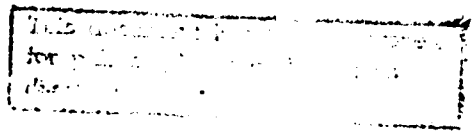
AN ERROR ANALYSIS TECHNIQUE FOR INERTIAL NAVIGATION SYSTEMS AND KALMAN FILTERS

by

C. E. Hutchinson and H. M. Wondergem

Contract No. ONR-N00014-68-A-0146-6
Report No. THEMIS-UM-68-2



September, 1968

Approved for Release

Reproduction in whole or in part is permitted for any purpose of the United States Government. In citing this manuscript in a bibliography, the reference should be followed by the phrase: UNPUBLISHED MANUSCRIPT.

Charles E. Carver

Charles E. Carver, Jr.
Co-Manager, Project THEMIS
University of Massachusetts

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

AA, A_1	an n x n control matrix
AC, C^*	an m x m covariance matrix of the measurement noise of the filter
AE	the absolute error used in the MEXP subroutine for the calculation of the transition matrices and error covariance matrices
AF, F^*	an n x n system matrix for the filter
AK, K_n	an n x m matrix for the filter gain
AM, M^*	an m x n measurement matrix of the states in the filter
AQ, Q^*	an n x n covariance matrix of the random driving terms in the filter
BA, A_2	an nb x n control matrix
BC, C	an m x m covariance matrix of the measurement noise of the system
BF, F	an nb x nb system matrix for the model of the system
BM, M	an m x nb measurement matrix of the states of the system
BPHI, Φ_n	an nb x nb transition matrix for the system
BQ, Q	an nb x nb covariance matrix of the random inputs of the system
BRK, R_n	an nb x nb covariance matrix of the system input error propagated one time step
COVU, $Cov(U_n)$	an nnb x nnb matrix of the covariance of U_n
COVZ, $Cov(z_n)$	an nnb x nnb matrix of the covariance of the augmented states
D	the time between updates, D = T/KT
g	the gravity of the earth
I	the identity matrix
IOUT	a flag used in subroutine OUT to select the print format of P1 and COVZ

KAI	a flag used in subroutine AINPUT to select the read format of the matrices in the data deck
KMAX	the maximum number of iterations in the MEXP subroutine
KT	the number of updates between measurements
KTF	the number of measurements to be taken
L	the latitude
M, m	the number of elements in the output vector
N, n	the number of states in the filter
NB, nb	the number of states in the system
NNB, nnb	the number of states in the augmented state vector z, nnb = n + nb
OH, Θ_n	an nnb x nnb "transition" matrix for the augmented states
P1, P_n	an n x n matrix for the error in estimation of the states in the filter
PHI, Φ_n^*	an n x n transition matrix for the filter
R	the radius of the earth
RE	the relative error used in subroutine MEXP for the calculation of the transition matrices and error covariance matrices
RK, R_n^*	an n x n covariance matrix of the filter input error term propagated one time step
S	an nb x n transformation matrix
T	the time between measurements
u_n	an nb-vector of the convolution of the random inputs, w, with the transition matrix
U_n	an nnb-vector of the convolution of the random inputs of the augmented states with the corresponding "transition matrix"
v(t)	an m-vector representing the random measurement errors
w(t)	an nb-vector representing the random inputs to the system
x	an nb-vector of the states in the system

x^*	an n-vector of the states in the system
$\hat{x}$	an n-vector of the estimates
$\tilde{x}$	an nb-vector denoting the error in estimation of the states in the system
y	an m-vector denoting the outputs of the system
z	an nnb-vector of the augmented states
β	the reciprocal of the correlation time of a random process
∇	the accelerometer errors
δR	the error in position, ft
$\delta \dot{R}$	the error in velocity, ft/sec
ε	the gyro errors (drift rates)
ν	the measurement bias error
ϕ_x, ϕ_y	the errors in platform tilt
ϕ_z	the error in azimuth
σ^2	the variance of a random process
Ω	the angular rate of rotation of the earth
ω	angular velocity

LIST OF TABLES

CHAPTER I

INTRODUCTION

A. Estimation in Optimum Control

Estimation techniques have become very important in modern control theory applications. In many cases the need for an estimation scheme arises when optimum control is to be applied to the system. The optimum control law is usually a function of all of the states of the system. However not all of the states are directly available. Therefore it is necessary to extract information concerning the states from the measurements which are available. This process is referred to as estimation.

A basic feature of any optimum estimation technique is that the estimate of the states is updated by continuously observing the output, so as to minimize some performance index which is generally a function of the error in estimation [1]. For the Kalman estimator [2, 3], used in this thesis, the variance of the error in estimation is minimized.

A block diagram of an estimation process is shown in Fig. 1. The estimator is employed to determine the optimum estimate of the state variables from the measured output variables. The controller is used to generate an optimum control law on the basis of the estimate of the state variables [4].

As in most problems, the optimum solution may not be the most economical or practical. Often it is not advantageous to estimate all of the state variables, rather only those states which have the largest influence on the control law. By doing this the complexity of the

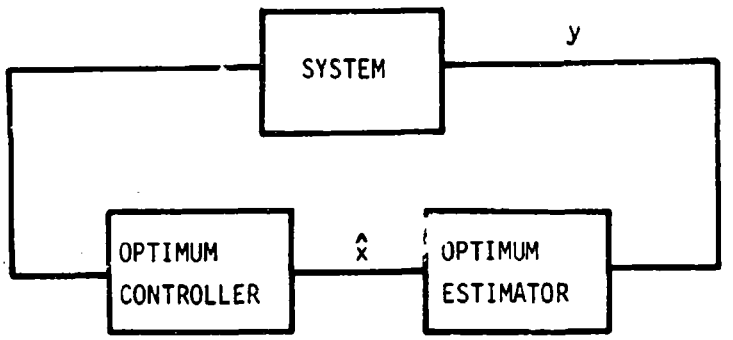


Fig. 1 Optimum estimation and control scheme.

estimator and the number of computations involved are reduced considerably. However the degree of accuracy will be less when compared to the optimum estimator.

B. Criteria for an Estimation Scheme

Various properties may be used as comparison criteria for an estimation scheme. The three criteria considered here are:

1. accuracy
2. speed of the computation
3. complexity.

All these criteria are closely related. To obtain a high accuracy, the technique will inevitably be a sophisticated and complex one. Complex techniques are limited however, because of the size of the available digital computer. The higher the required accuracy, the more complex the technique has to be and the bigger and more expensive the computer has to be. Besides accuracy, a second important factor to be considered is the time between measurements. The time it takes to calculate a new estimate

depends on the complexity of the technique and the speed of the computer. Obviously the smaller this time the better. When the measurement time decreases, the discrete case approaches the continuous case, and more information about the states becomes available in a shorter time, yielding a possibility that the system reaches its steady state much faster.

In many cases such as airborn navigation systems, measurements must be taken at short intervals because of the speed of the aircraft and the short duration of the flight. In marine navigation systems the speed of computation is not critical and often accuracy appears to be less critical too. All considerations call for an estimator which is as simple as possible to implement on a special purpose digital computer, i.e., an estimator that estimates only those states which are necessary to obtain a good control. Though the estimator may not be optimum anymore, and the accuracy decreased, the time of computation may be decreased sufficiently to allow more measurements. This simplification of the estimator results in a smaller computer, and possibly an increased number of measurements which makes up for the lost accuracy, as more information can be obtained in a shorter time. The trade-off between time of calculations and the complexity of the computer, however, has to be considered for every specific case.

This thesis develops a computer program with which it is possible to simulate dynamical systems described by linear differential equations with constant coefficients and random driving terms, and rapidly to compare and to analyze different estimation techniques with respect to accuracy and computational complexity.

Since the equations of the system will be written in state variable

form, vector notation will be used throughout. Although the estimator equations can be written in continuous as well as discrete form, only the discrete equations are considered since a digital computer simulation will be used.

CHAPTER II

MATHEMATICAL SOLUTION TO THE PROBLEM

A. Formulation of the Problem

In this chapter the necessary equations are presented that provide a framework by which it is possible to study the performance of the optimum and sub-optimum Kalman estimator or, as it is sometimes called, the Kalman filter. The differential equation describing the system is

$$\dot{x}(t) = F\,x(t) + w(t)$$
$$y(t) = M\,x(t) + v(t) \qquad (2\text{-}1)$$

where

- $x(t)$ an nb-vector* denoting the states in the system, with initial condition $x(0) = x_0$.
- $w(t)$ an nb-vector of gaussian, white noise processes with zero mean
- $y(t)$ an m-vector of the outputs of the system
- $v(t)$ an m-vector of the errors (gaussian, white noise sequence)
- F an nb x nb system matrix
- M an m x nb measurement matrix.

The number of states in the system is given by nb, and the number of outputs by m.

To solve the problem numerically, Eq. (2-1) can be rewritten as a difference equation

$$x_n = \Phi_n\,x_{n-1} + u_n \qquad (2\text{-}2)$$

*All vectors are considered to be column vectors unless otherwise indicated.

where

$$u_n = \int_{t_{n-1}}^{t_n} \phi(t_n, \tau)\, w(\tau)\, d\tau \qquad (2\text{-}3)$$

where $\phi_n = \phi(t_n, t_{n-1})$ is the transition matrix, describing the change of the state vector from time t_{n-1} to time t_n. Since w(t) and v(t) are random, x(t) and y(t) will also be random. Therefore when an estimate of x(t) is generated based upon the measurements y(t), this quantity will be random. The approach that will be utilized to study these random quantities will be to investigate their covariance matrices. That is the matrix defined by

$$\text{Cov}\,[x(t)] = E[x(t)x^T(t)] \qquad (2\text{-}4)$$

where superscript T denotes matrix transposition and E is the expectation operator. In general, w(t) and v(t) have zero mean, making x(t) and y(t) have zero mean values. Statistically the covariances of w(t) and v(t) can be described by

$$E[w(t)w^T(\tau)] = Q\delta(t - \tau) \qquad (2\text{-}5)$$

$$E[v(t_i)v^T(t_j)] = C\delta_{ij} \qquad (2\text{-}6)$$

where $\delta(t - \tau)$ is a Dirac delta function for continuous signals and δ_{ij} is the Kronecker delta function for discrete signals.

B. Optimum Kalman Filter

The optimum Kalman filter estimates all the states of the system, and uses the same mathematical model for the filter as for the system. A block diagram of the optimum Kalman estimator is given in Fig. 2. Defining the states of the system by x(t) and the estimate of the state

by $\hat{x}(t)$, the error in estimation is defined by

$$\tilde{x}(t) = x(t) - \hat{x}(t)$$

If an estimate is provided at some time t_{n-1}, and no subsequent measurements are available, the best estimate of the states at all subsequent times is obtained by solving the deterministic portions of Eq. (2-2)

$$\hat{x}_n = \Phi_n \hat{x}_{n-1} \tag{2-7}$$

where Φ_n is the same transition matrix used for updating the states of the system. The estimate at t_{n-1} is used as an initial condition. At time t_n the errors in estimation will have propagated according to

$$x_n - \hat{x}_n = \Phi_n x_{n-1} + u_n - \Phi_n \hat{x}_{n-1}$$

or

$$\tilde{x}_n = \Phi_n \tilde{x}_{n-1} + u_n \tag{2-8}$$

The resulting value for $\tilde{x}_n$ can be used as the initial condition for the following update. However, if a measurement is obtained at time t_n, a new estimate can be calculated. The use of measurements provided at discrete instants of time causes the error covariance to be discontinuous, having different values before and after the measurements. For this reason, the error immediately before a measurement is designated $\tilde{x}(-)$ and the same error after the measurement is $\tilde{x}(+)$. The same notation applies to the estimate and other matrices with a discontinuity at the measurement time.

With the new estimate, the error in estimation becomes

$$\tilde{x}_n(+) = x_n - \hat{x}_n(+)$$

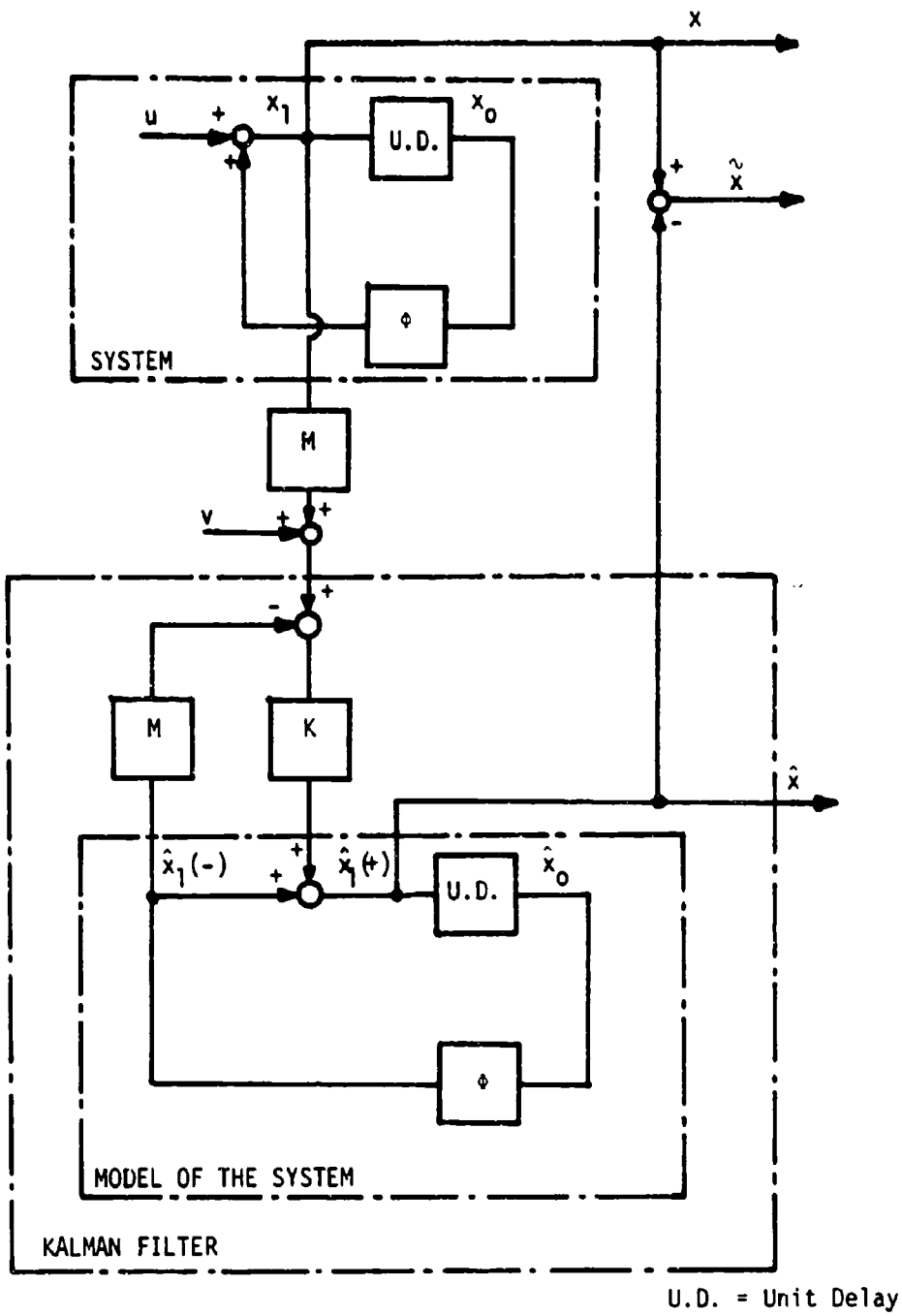


Fig. 2 Optimum Kalman estimation scheme.

Hopefully the new estimate will cause the error, or rather its covariance to be reduced in some way.

To solve the problem statistically, the covariance is taken of Eq. (2-8)

$$\text{Cov}\,(\tilde{x}_n) = \phi_n\,\text{Cov}\,(\tilde{x}_{n-1})\,\phi_n^T + \text{Cov}\,(u_n) \tag{2-9}$$

Using the notation

$$R_n = \text{Cov}\,(u_n) \quad \text{and} \quad P_n = \text{Cov}\,(\tilde{x}_n)$$

Eq. (2-6) can be written as

$$P_n = \phi_n\,P_{n-1}\,\phi_n^T + R_n \tag{2-10}$$

The techniques which yield numerical solutions for ϕ_n and R_n are derived in Appendix A, and are given by

$$\phi_n = e^{F(t_n - t_{n-1})} = \sum_{i=0}^{\infty} \frac{F^i (t_n - t_{n-1})^i}{i!} \tag{2-11}$$

$$R_n = \sum_{i=1}^{\infty} Q_n \frac{(t_n - t_{n-1})^i}{i!} \tag{2-12}$$

where Q_n is given by

$$Q_n = FQ_{n-1} + (FQ_{n-1})^T \tag{2-13}$$

The initial condition for Q_n is given by the covariance matrix for w(t), see Eq. (2-5). The covariance matrix of the error in estimation is at discrete time instances sequentially updated according to Eq. (2-10). This simulates the dynamic behavior of the covariance of the error in estimation between the measurements, which can now be observed each time the error is updated. The covariance of the error in estimation will propagate according to Eq. (2-10) until the time when a measurement is taken and a new estimate is calculated. The updated estimate is given

by

$$\hat{x}_n(-) = \Phi_n \hat{x}_{n-1}(+) \tag{2-14}$$

By taking a measurement a correction term $\Delta\hat{x}_n$ is found which gives a new estimate

$$\hat{x}_n(+) = \hat{x}_n(-) + \Delta\hat{x}_n \tag{2-15}$$

According to Ref. [3] the equations to calculate $\Delta\hat{x}_n$ can be written as

$$\Delta x_n = K_n[y_n - M\hat{x}_n(-)] \tag{2-16}$$

where

$$K_n = P_n(-)\, M^T[M\, P_n(-)\, M^T + C]^{-1} \tag{2-17}$$

and

$$y_n = M\, x_n + v$$

Having found the optimum gain K_n, the new estimate is calculated according to Eqs. (2-15) and (2-16).

$$\hat{x}_n(+) = \hat{x}_n(-) + K_n[y_n - M\, \hat{x}_n(-)] \tag{2-18}$$

The new covariance matrix of the error is found by taking the covariance of Eq. (2-18) yielding

$$P_n(+) = [I - K_n\, M]\, P_n(-)\, [I - K_n\, M]^T + K_n\, C\, K_n^T \tag{2-19}$$

This equation is usually used in its simpler but numerically equivalent form (see Ref. [5])

$$P_n(+) = [I - K_n\, M]\, P_n(-) \tag{2-20}$$

The only drawback of Eq. (2-20) is that algorithmically the equation does not yield a symmetric matrix. Round off errors in the computer cause the matrix to be non-symmetric resulting in errors which can

propagate rapidly, especially with a large number of states.

To analyze the accuracy of the optimum Kalman filter it is only necessary to study the time behavior of P_n. The equations for this time history are given by Eqs. (2-10), (2-17) and (2-20). The flow diagram of an analysis program for this purpose is shown in Fig. 3.

C. Sub-optimum Kalman Filter with Control Matrices

In case of a sub-optimum filter, the dynamical model used to update the estimate is different from the dynamical model of the system. Generally, the transition matrix in the filter is of a lower order than the transition matrix of the system. This is in order to simplify and reduce the computations involved in calculating a new estimate. The matrices used in the filter will be specified with an asterisk.

The differential equation used in the filter to simulate the system is given by

$$\dot{x}^*(t) = F^* x^*(t) + w^*(t) \tag{2-21}$$

which is in discrete form written as

$$x_n^* = \phi_n^* x_{n-1}^* + u_n^* \tag{2-22}$$

The model of the output process is

$$y_n = M^* x_n^* + v^* \tag{2-23}$$

The new estimate calculated at the measurement time is therefore given by

$$\hat{x}_n(+) = \hat{x}_n(-) + K_n[y_n - M^* \hat{x}_n(-)] \tag{2-24}$$

where

$$\hat{x}_n(-) = \phi_n^* \hat{x}_{n-1}(+) \tag{2-25}$$

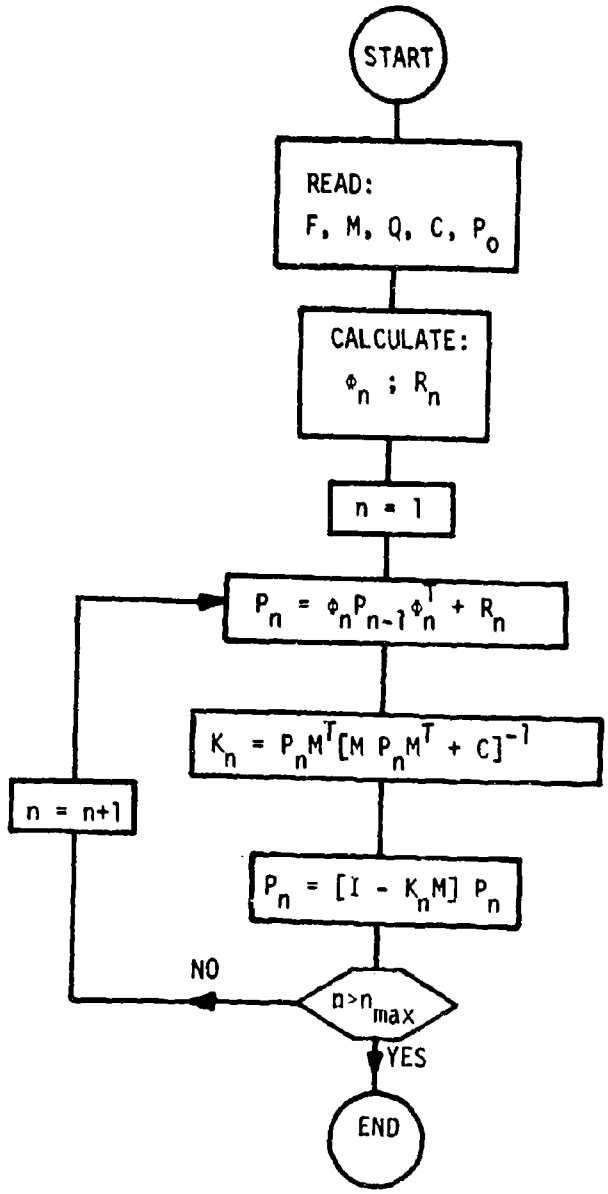


Fig. 3 Flow diagram of an optimum Kalman estimation program.

The equations for the sub-optimum estimator are in analogy to Eqs. (2-11), (2-17) and (2-20) given by

$$P_n(-) = \Phi_n^* P_{n-1}(+) \Phi_n^{*T} + R_n^* \tag{2-26}$$

$$K_n = P_n(-) M^{*T}[M^* P_n(-) M^{*T} + C^*]^{-1} \tag{2-27}$$

$$P_n(+) = [I - K_n M^*] P_n(-) \tag{2-28}$$

In the equations above, C^* is the covariance matrix of the modeled measurement error v^* (Eq. 2-6), which may or may not be the same as C, the true covariance. Likewise, there is a Q^* used in the calculation of R_n^* (Eq. 2-5). It should also be noted that with the sub-optimum filter P_n is no longer the true covariance matrix of the error in estimation. This fact can be seen using the following development.

To transform the estimate $\hat{x}$ into a vector with the same dimension as the state vector x, a transformation matrix S is required, such that

$$\tilde{x} = x - S \hat{x} \tag{2-29}$$

The new vector $S \hat{x}$ represents the estimate of all the states of the system. The covariance matrix of the error between the system states and the estimated states is

$$\text{Cov}(\tilde{x}) = E[(x - S \hat{x})(x - S \hat{x})^T]$$

or

$$\text{Cov}(\tilde{x}) = E(xx^T) - E(x\hat{x}^T)S^T - S E(\hat{x}x^T) + S E(\hat{x}\hat{x}^T)S^T \tag{2-30}$$

When solving this equation it is convenient to augment the state vector x by the estimation vector $\hat{x}$, which yields a new column vector z defined by*

$$z = \begin{bmatrix} x \\ \hat{x} \end{bmatrix}$$

*The vector z will be referred to in the following sections as the "augmented state vector."

The reason for doing this becomes obvious when the covariance of z is taken.

$$\text{Cov}(z) = E\left[\begin{array}{c|c} xx^T & x\hat{x}^T \\ \hline \hat{x}x^T & \hat{x}\hat{x}^T \end{array}\right] \tag{2-31}$$

By merely partitioning the Cov(z) matrix the auto-correlations and cross-correlations of x and $\hat{x}$ are directly defined. If Cov(z) is known, the covariance of the error can be calculated by using Eq. (2-30).

However there is another method [6] which does not require the partitioning of the Cov(z) matrix. Use is made of two control matrices A_1 and A_2. With these matrices control is applied to the system at the measurement times, according to the following equations

$$\hat{x}_n'(+) = \hat{x}_n(+) - A_1\,\hat{x}_n(+) \tag{2-32}$$

$$x_n' = x_n - A_2\,\hat{x}_n(+) \tag{2-33}$$

where $\hat{x}_n(+)$ is the new calculated estimate at the measurement time before control has been applied. The vectors x_n' and $\hat{x}_n'(+)$ are respectively the system state, and the estimate after control has been applied.

To simulate and observe the dynamic behavior of both the state of the system and the estimation of the state, the covariance matrix of z must be updated at discrete intervals. The covariance of the state is given by the matrix in the upper left corner of the Cov(z) matrix in Eq. (2-31), and the covariance matrix of the estimate is in the lower right corner. From Eqs. (2-2) and (2-25) it is easily derived that between measurements the relation of z_n to z_{n-1} is described by

$$z_n = \left[\begin{array}{c|c} \Phi_n & 0 \\ \hline 0 & \Phi_n^* \end{array}\right] z_{n-1} + \left[\begin{array}{c} u_n \\ \hline 0 \end{array}\right] \tag{2-34}$$

This equation can be written in the form

$$z_n = \Theta_n \, z_{n-1} + U_n \tag{2-35}$$

where Θ_n is a transition matrix for the augmented state and U_n is an error vector due to the random driving terms. Taking the covariance of Eq. (2-35) yields

$$\mathrm{Cov}(z_n) = \Theta_n \, \mathrm{Cov}(z_{n-1}) \; \Theta_n^T + \mathrm{Cov}(U_n) \tag{2-36}$$

where from Eq. (2-34) and (2-11) $\mathrm{Cov}(U_n)$ is found to be

$$\mathrm{Cov}(U_n) = \left[\begin{array}{c|c} R_n & 0 \\ \hline 0 & 0 \end{array}\right]$$

At the measurement time a new "optimum" gain K_n is calculated with Eq. (2-27), and a new matrix $P_n(+)$ is obtained from Eq. (2-28). Having calculated a new gain K_n the estimate is corrected according to

$$\hat{x}_n(+) = \hat{x}_n(-) + K_n[y_n - M^* \, \hat{x}_n(-)] \tag{2-37}$$

where y_n is given by the system equation

$$y_n = M \, x_n + v$$

The new estimate $\hat{x}_n(+)$ is then fed back through the control matrices to x_n and $\hat{x}_n(+)$ as described in Eqs. (2-32) and (2-33). By substituting Eq. (2-37) into these two equations the new values for the state and the estimate are

$$x_n' = (I - A_2 \, K_n \, M)x_n - A_2(I - K_n \, M^*) \, \hat{x}_n(-) - A_2 \, K_n \, v \tag{2-38}$$

$$\hat{x}_n'(+) = (I-A_1)K_n M x_n + (I-A_1)(I - K_n M^*)\hat{x}_n(-) + (I-A_1)v \qquad (2\text{-}39)$$

where the prime denotes that control has been applied. Augmenting x_n' with $\hat{x}_n'$ will give the new value for z_n'. The equation describing the relation between z_n' and z_n is of the same form as Eq. (2-35)

$$z_n' = \Theta_n' z_n + U_n' \qquad (2\text{-}40)$$

where Θ_n' and U_n' are given by

$$\Theta_n' = \left[\begin{array}{c|c} I - A_2 K_n M & -A_2(I - K_n M^*) \\ \hline (I - A_1)K_n M & (I - A_1)(I - K_n M^*) \end{array}\right] \qquad (2\text{-}41)$$

$$U_n' = \left[\begin{array}{c} -A_2 K_n v \\ \hline (I - A_1)v \end{array}\right] \qquad (2\text{-}42)$$

Taking the covariance matrix of Eq. (2-40) yields

$$\mathrm{Cov}(z_n') = \Theta_n' \,\mathrm{Cov}(z_n)\, \Theta_n'^{T} + \mathrm{Cov}(U_n') \qquad (2\text{-}43)$$

where

$$\mathrm{Cov}(U_n') = \left[\begin{array}{c|c} A_2 K_n C K_n^T A_2^T & -A_2 K_n C K_n^T (I-A_1)^T \\ \hline -(I-A_1)K_n C K_n^T A_2^T & (I-A_1)K_n C K_n^T (I-A_1)^T \end{array}\right]$$

which can also be written in the simpler form

$$\mathrm{Cov}(U_n') = \left[\begin{array}{c} -A_2 \\ \hline I - A_1 \end{array}\right] K_n C K_n^T \left[-A_2^T \,\vdots\, I - A_1\right] \qquad (2\text{-}44)$$

D. Function of the Control Matrices

The function of the control matrices is shown in Fig. 4. With A_1 and A_2 equal to zero there will be no control, and the normal propagation of the state vector x and the estimate $\hat{x}$ is simulated. With the option of having A_1 and A_2 it is possible to propagate the covariance matrix of

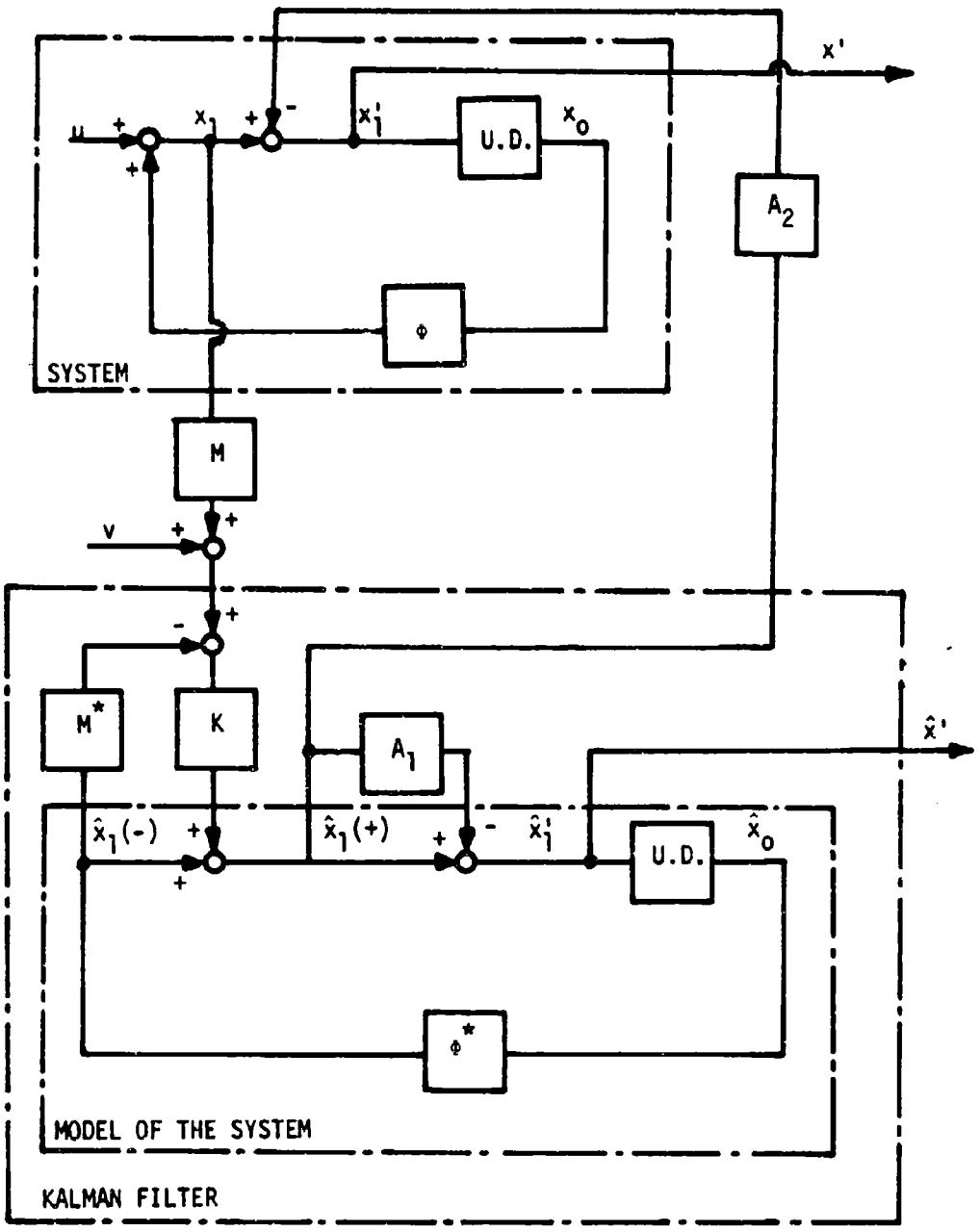


Fig. 4 Sub-optimum estimation scheme with control matrices.

the error in estimation

$$\mathrm{Cov}(\tilde{x}) = \mathrm{Cov}(x - S\,\hat{x})$$

Making $A_1 = I$ and $A_2 = S$, the state vector and the estimate become

$$x_n' = x_n - S\,\hat{x}_n(+) = \tilde{x}_n(+) \tag{4-45}$$

$$\hat{x}_n'(+) = (I - A_1)\hat{x}_n(+) = 0 \tag{4-46}$$

In other words, the vector z_n' contains the error in estimation in its upper part. Updating the covariance of z_n' will simulate the propagation of the error in estimation.

The advantage of this technique is that it is not necessary anymore to calculate the covariance of the error separately from the updating of $\mathrm{Cov}(z_n)$. With the option of the two control matrices, the covariance of the error is obtained, in the upper left corner of $\mathrm{Cov}(z_n)$ as indicated above by making $A_1 = 0$ and $A_2 = S$. A block diagram for a sub-optimum filter is contained in Fig. 4. A flow diagram for an analysis of a sub-optimum filter is contained in Fig. 5.

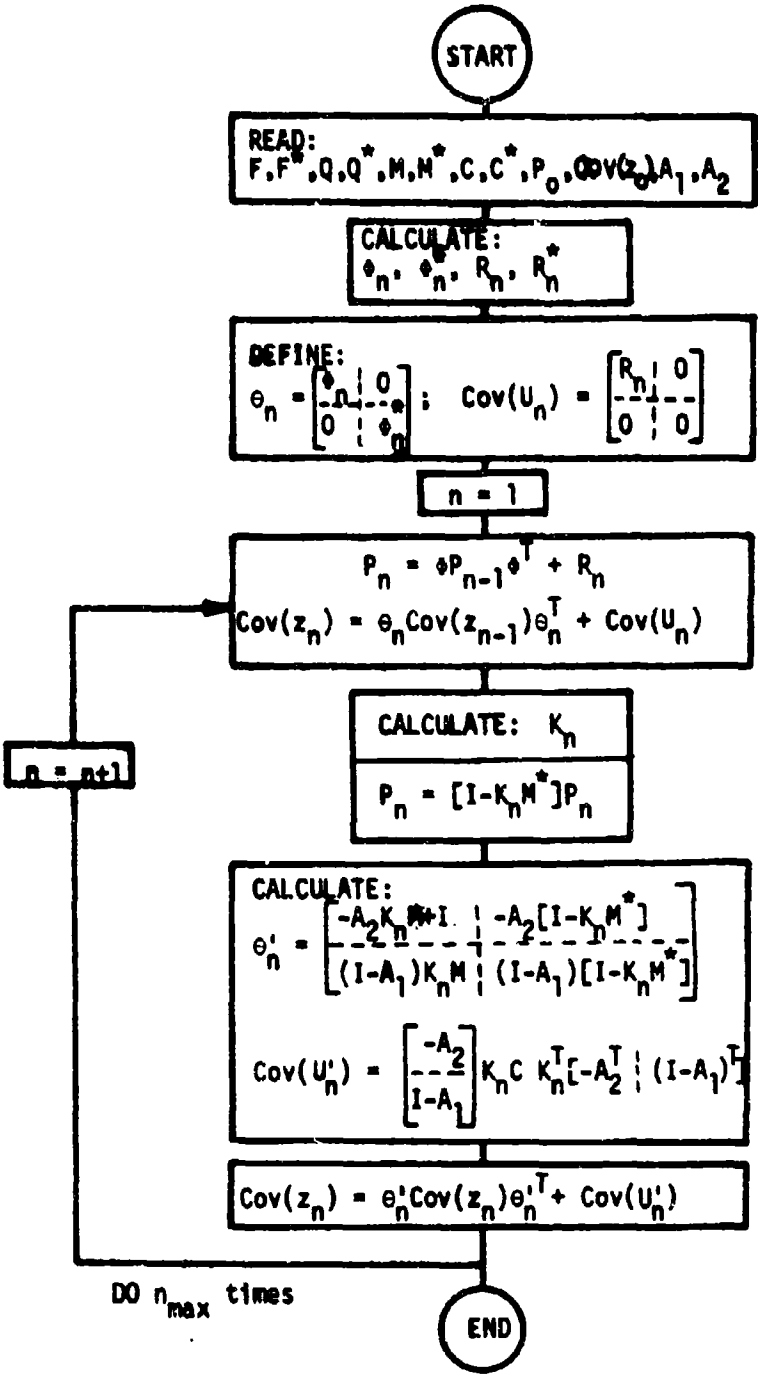


Fig. 5 Flow diagram of a sub-optimum Kalman estimator.

CHAPTER III

DESCRIPTION OF THE PROGRAM

A. Introduction

This chapter contains an explanation of a computer program developed for analyzing estimation schemes with a Kalman filter, using state variable techniques. The program utilizes the equations derived in Chapter II for the sub-optimum Kalman estimator with control matrices. A complete listing of the program is contained in the Appendices.

The program was written in Fortran IV for the CDC 3600. Variable dimensions are used throughout the program except for the dummy arrays, which are used only for temporary storage to perform certain matrix operations. These temporary matrices denoted by T1, T2, etc., are in COMMON. This is done because every subroutine requires a different number of dummy arrays of different sizes. The COMMON can then be arranged to suit the need of every subroutine. All other arrays have variable dimensions, which makes it convenient to change the storage assigned to these matrices by only changing the DIMENSION statement in the main program. Changing dimensions becomes important when the memory space available in the computer is limited. Certain constant matrices which do not need many computations to compute, are recalculated with every measurement in order to use their storage space for the calculation of other matrices. This way, without losing much time a great deal of memory space is obtained. With this program it is possible on a computer with 32k memory locations to analyze estimation schemes with up to 25

states, 25 estimates and 10 outputs, or any other combination which results in an equal amount of storage. However, the DIMENSION statement in the main-line program (and the COMMON) has to be changed accordingly. The different subroutines used in the program and how they are linked together are shown in Fig. 6.

B. Data Deck Setup

The main-line program called ESTIM, starts by reading all the non-dimensioned variables used in the program. These variables are to be punched on the first data card which will be read according to the following read statements

```
1  FORMAT(8I5,3E10.5)
   READ 1, N,NB,M,KT,KTF,KMAX,IOUT,KAI,T,AE,RE
```

where

N the number of states in the filter

NB the number of states in the system

M the number of elements in the output

KT the number of updates between measurements

KTF the number of measurements to be taken

KMAX the maximum number of iterations allowed in the MEXP subroutine for the calculation of the transition matrices and the random error term of the matrix Ricatti equations

IOUT a flag used in the print routine OUT, and set equal to:
1 when the square root of only the diagonal terms of P1 and COVZ are to be printed as output, or
2 when the full matrix of P1 and COVZ are to be printed out.

KAI a flag which is used in the subroutine AINPUT, and set equal to:
1 when only the diagonal terms of AQ, BQ, P1, and COVZ are used as input
2 when the diagonal and sub-diagonal terms of these symmetric matrices are to be read

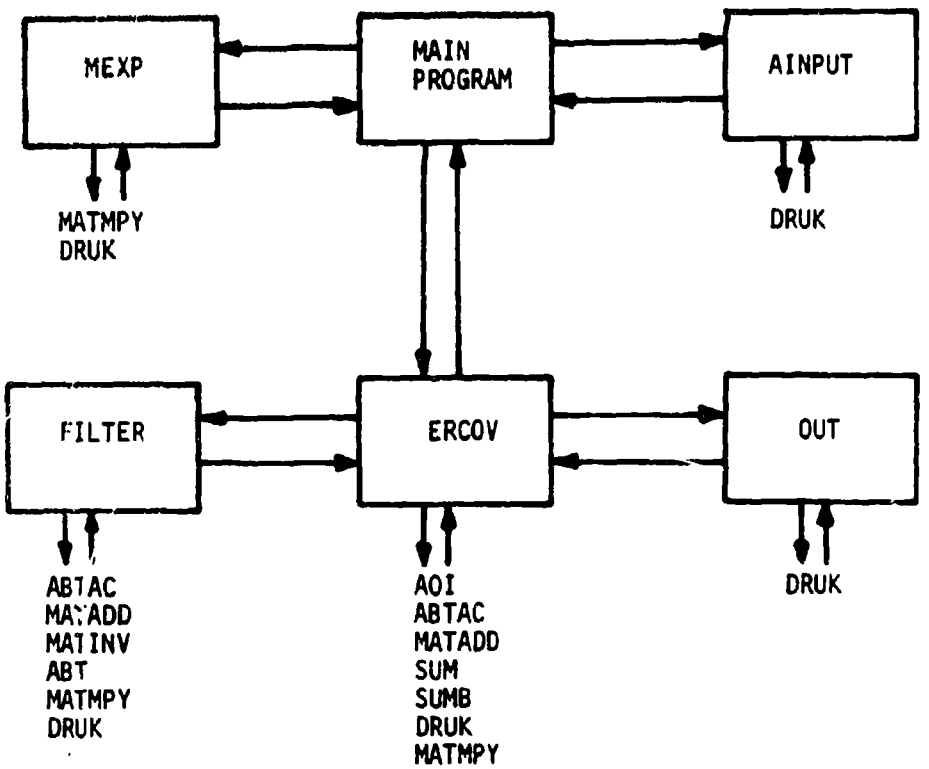


Fig. 6 Block diagram of the subroutines used in the program.

T the time between measurements, in the same time unit as used in the system matrices

AE and RE are respectively the absolute and the relative error which are allowed in the calculation of PHI, BPHI, RK, BRK

The next set of data cards contain sequentially the following matrices

AF an n x n matrix which is used as model of the system in the filter

BF an nb x nb matrix representing the model of the system

AQ an n x n covariance matrix of the random driving terms in the model of the filter

BQ an nb x nb covariance matrix of the random driving terms in the model of the system

AM an m x n measurement matrix of the states in the filter

BM an m x nb measurement matrix of the states in the system

AC an m x m covariance matrix of the noise at the output of the filter

BC an m x m covariance matrix of the noise at the output of the system

P1 an n x n covariance matrix of the error in estimation

COVZ an (nb+n) x (nb+n) covariance matrix of the augmented vector of the states in the system and the estimated states in the filter

AA an n x n control matrix feeding back the estimate after the measurement according to Eq. (2-32)

BA an nb x n control matrix feeding back the estimate after the measurement to the states of the system (see Eq. 2-33).

C. Input Formats

The matrices are read in row-wise according to FORMAT(8E10.4). If the number of elements in a row is greater than the number of fields specified by the format statement, i.e., eight in this case, the reading of the elements is continued on the following data card(s). The next

row is started at a new data card, which continues until all rows are read. An example of the statements used for reading a matrix A with dimensions n x m is given by

```
1  FORMAT(8E10.4)
   DO 9 I=1,N
9  READ 1,(A(I,J),J=1,M)
```

The reading of AQ, BQ, P1, COVZ is slightly different. Here advantage is taken of the fact that these matrices are symmetric. Therefore only the elements of the lower triangle including the elements on the diagonal are read. Again the reading is done row-wise as was the case with a non-symmetric matrix. The only difference is that now the diagonal term is considered to be the last element in the row. Inside the subroutine AINPUT the upper-diagonal terms are equated to their corresponding sub-diagonal terms. The statements to read a symmetric matrix B with dimension n x n are

```
1  FORMAT(8E10.4)
   DO 8 I=1,N
8  READ 1, (B(I,J),J=1,I)
```

A third type of matrix is the diagonal matrix AA. The diagonal elements are put in order on the same data card, or subsequent data cards if the number of elements of the diagonal is greater than 8. All off-diagonal terms are set to zero inside the subroutine AINPUT. The statements used for reading a diagonal matrix C with dimensions n x n are

```
1  FORMAT(8E10.4)
   READ 1, (C(I,I),I=1,N)
```

In most cases the off-diagonal terms of AQ, BQ, P1 and COVZ are also equal to zero. The flag KAI provides the option to read only the

diagonal terms of these matrices, in the same way as for AA.

As a final check if the data was presented and read correctly, all the variables and matrices that are read as input data, are directly printed. The printing of the matrices is done in a separate subroutine DRUK, so that the print format can easily be changed.

D. General Description of the Program

After all the data is read and printed, the transition matrices and the error covariance matrices are calculated in the subroutine MEXP. The accuracy desired for these matrices is to be defined by AE and RE, the absolute and relative error terms respectively. The iterations continue until all elements in the matrix EXPTA satisfy the inequality

$$|\Delta EXPTA_{ij}| < AE + RE\ |EXPTA_{ij}| \qquad (3\text{-}1)$$

where EXPTA is the desired matrix and ΔEXPTA is the last calculated term of the series. When Eq. (3-1) applies for each element of ΔEXPTA, the total number of iterations is printed and the final matrix EXPTA is printed by calling DRUK. KMAX provides a limit on the number of iterations allowed. In the case a matrix does not converge within the allowable number of iterations, an error message will be printed and further execution of the program terminated.

Following the calling of MEXP, control is transferred to subroutine ERCOV where the matrices P1 and COVZ are updated and corrected at each measurement. The way the system and the filter are updated and measurements are taken is illustrated in Fig. 7. The initial conditions for P1 and COVZ are specified by the input data. The first update is at t = T/KT, the states in the filter and the system are updated without the correction

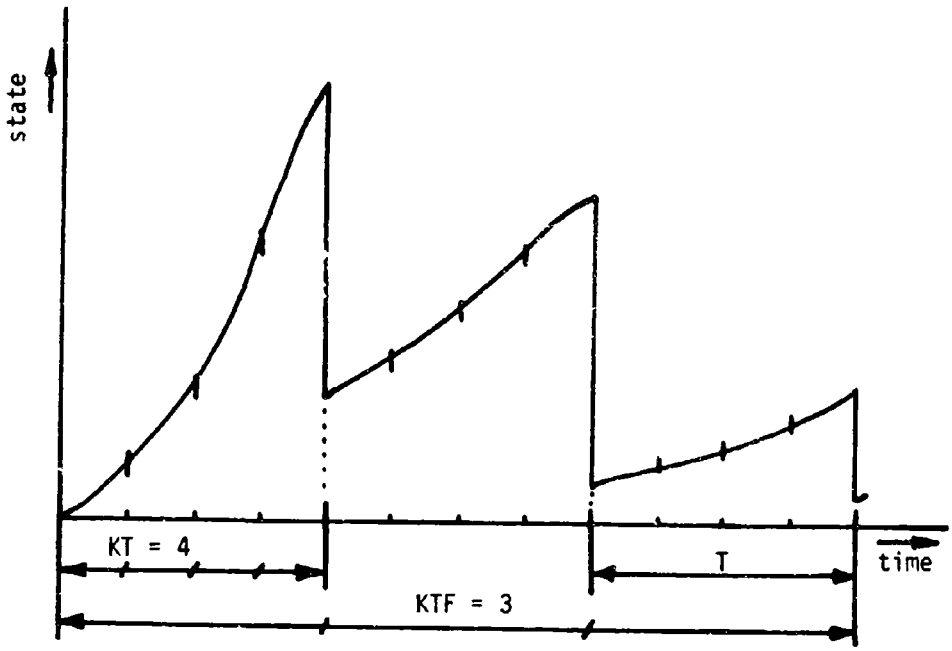


Fig. 7 Propagation of the states, with measurements.

of the estimate by the Kalman filter. The number of updates does not have any effect on the propagation of the states; it merely provides a possibility to observe the states at certain times between measurements. At every update the corresponding time is printed, and P1 and COVZ are printed in subroutine OUT. According to the value of the flag IOUT, P1 and COVZ are either printed completely or the square root of only the diagonal terms is printed.

After KT updates have been calculated and printed out, the time will be t = T, which is equal to the measurement time. The number of the measurement is printed and the control is transferred to subroutine FILTER. Here, the new optimum gain AK is calculated according to Eq. (2-27) and printed. With the new gain, P1 is corrected according to Eq. (2-28). When control returns, COVZ is implemented with the new estimate as described by Eq. (2-43). The new P1 and COVZ are printed

again in subroutine OUT. Control returns to the beginning of ERCOV, and the procedure of updating and taking a measurement continues. After KTF measurements have been taken, control returns to the main-line program.

The program is implemented with a DO-loop to accept up to ten data decks. An "END OF FILE" (EOF) check is performed and execution of the program is terminated when the EOF card is encountered in the case of less than ten data decks. The first card of the new data deck follows directly after the last card of the preceding data deck.

E. Explanation of the Subroutines

ESTIM: The main-line program, ESTIM, is kept quite simple. A DO-loop with a dummy variable, II, gives the possibility of accepting up to ten data decks. Inside the DO-loop, the first data card with all the variables is read as explained in the beginning of this chapter. With the following statement, CALL AINPUT, all input matrices are read. After control returns, the subroutine MEXP is called four times: twice with the flag KLM = 1, to calculate the transition matrices PHI and BPHI, respectively of the filter and the system; and two times with the flag KLM = 2, for the calculation of the error covariance terms of the filter and the system, designated RK and BRK. These call statements are followed by a call statement for subroutine ERCOV where the rest of the calculations are performed. After the execution of this statement, control goes back to the beginning of the DO-loop and reads the first data card of the next data deck. With the "END OF FILE" check following the read statement, the program is terminated when the EOF

card is encountered in the case of less than ten data decks.

Subroutine AINPUT: In this subroutine the following matrices are read: AF, BF, AQ, BQ, AM, BM, AC, BC, P1, COVZ, AA, BA according to the format explained in the beginning of this chapter. This subroutine is called with

AINPUT(AF,BF,AQ,BQ,AM,BM,AC,BC,P1,COVZ,AA,BA,N,NB,M,NNB,KAI)

where KAI is the flag to select the read format of the matrices and NNB is the number of states in the augmented state vector, z.

Subroutine MEXP: Referring to the matrix exponential and the error covariance flow diagrams in Appendix A, it can be noted that a large similarity exists between both diagrams. Consequently, both programs have been combined into one called MEXP. The flow diagram of this subroutine is shown in Fig. 8. According to the value of the flag KLM, either the equations for the matrix exponential or the error covariance routines are used. When KLM = 1 the transition matrix is computed and with KLM = 2 the error covariance matrix is calculated. Both matrices are required to update the covariance matrix of the states according to Eq. (2-26).

In order to be able to acquire a high accuracy with slowly converging matrices many iterations are necessary. With large matrices this involves many computations which result in an undesirable growth of the truncation errors in the computer. Double precision is used for the algorithm to circumvent this problem. The parameter list in this subroutine is

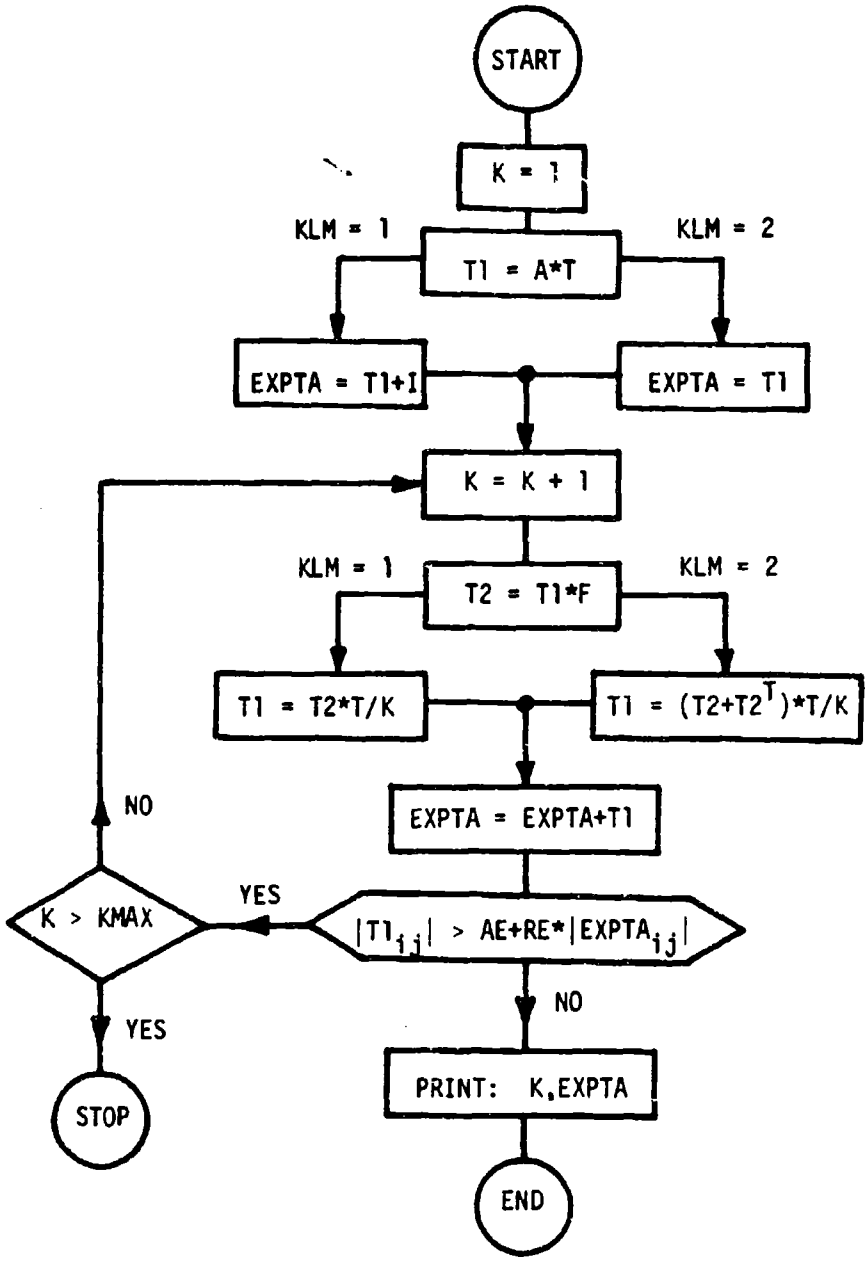


Fig. 8 Flow diagram of the MEXP subroutine for the numerical solution of the matrix exponential and error covariance equations.

```
MEXP(D,F,A,T1,T2,EXPTA,T3,N,KMAX,AE,RE,KLM)
```

The following variables are to be specified in the calling program

D the time between updates, D = T/KT

F the system matrix

A either the system matrix F, when KLM = 1, or the covariance of the random driving terms Q, when KLM = 2.

N the order of the matrices

KMAX the maximum number of iterations

AE the absolute error

RE the relative error

KLM a flag

T1, T2 and EXPTA are double precision arrays which are used only for temporary storage inside the subroutine. EXPTA is the desired matrix in double precision which at the end of all computations is equated with the single precision matrix T3 which is transferred to the calling program through the parameter list. The call statement for calculating a transition matrix PHI might, for example, be

```
CALL MEXP(D,F,F,T1,T2,T3,PHI,N,KMAX,AE,RE,1)
```

The equivalent statement for the calculation of the error covariance term RK would be

```
CALL MEXP(D,F,Q,T1,T2,T3,RK,N,KMAX,AE,RE,2)
```

Generally the subroutine operates as follows: The single precision variable D is equated to the double precision variable T. Next, the statements for EXPTA = T1 = A*T are executed. For KLM = 1 the identity matrix I is added to this first term of the series. The double precision

variable K, denoting the number of iterations, is incremented by 1. The following term of either series is calculated and stored in T1 and added to the series. All elements of T1 are checked according to Eq. (3-1). A check follows to determine if the number of iterations K, has exceeded the limit KMAX. If one of the elements of T1 does not satisfy Eq. (3-1) and K is still below its limit, the iterations continue and the next term of the series is calculated. In the case that the series does not converge and the number of iterations reaches its limit KMAX, the message "number of iterations exceeded" will be printed and the execution of the program terminated.

Subroutine ERCOV: In this subroutine P1 and COVZ are updated and after each measurement incremented with a new estimate. The flow diagram of ERCOV is represented in Fig. 9. The parameter list in the subroutine is given by

```
ERCOV(PHI,BPHI,RK,BRK,AM,BM,AK,AC,BC,P1,OH,COVU,COVZ,AA,BA,TA,TC,
N,NB,M,NNB,KT,KTF,IOUT,D)
```

Except for AK, OH, COVU, TA and TC all variables and arrays are to be specified in the calling program. The matrices which have not yet been defined are

OH the transition matrix of the augmented state vector z

COVU the error covariance term for the augmented z-vector

TA and TC two dummy arrays with variable dimension.

The subroutine starts with storing of AA - I in AA, where I is the identity matrix. This is done because AA appears always in conjunction with I. After these steps, control comes to the first DO-loop where KTF

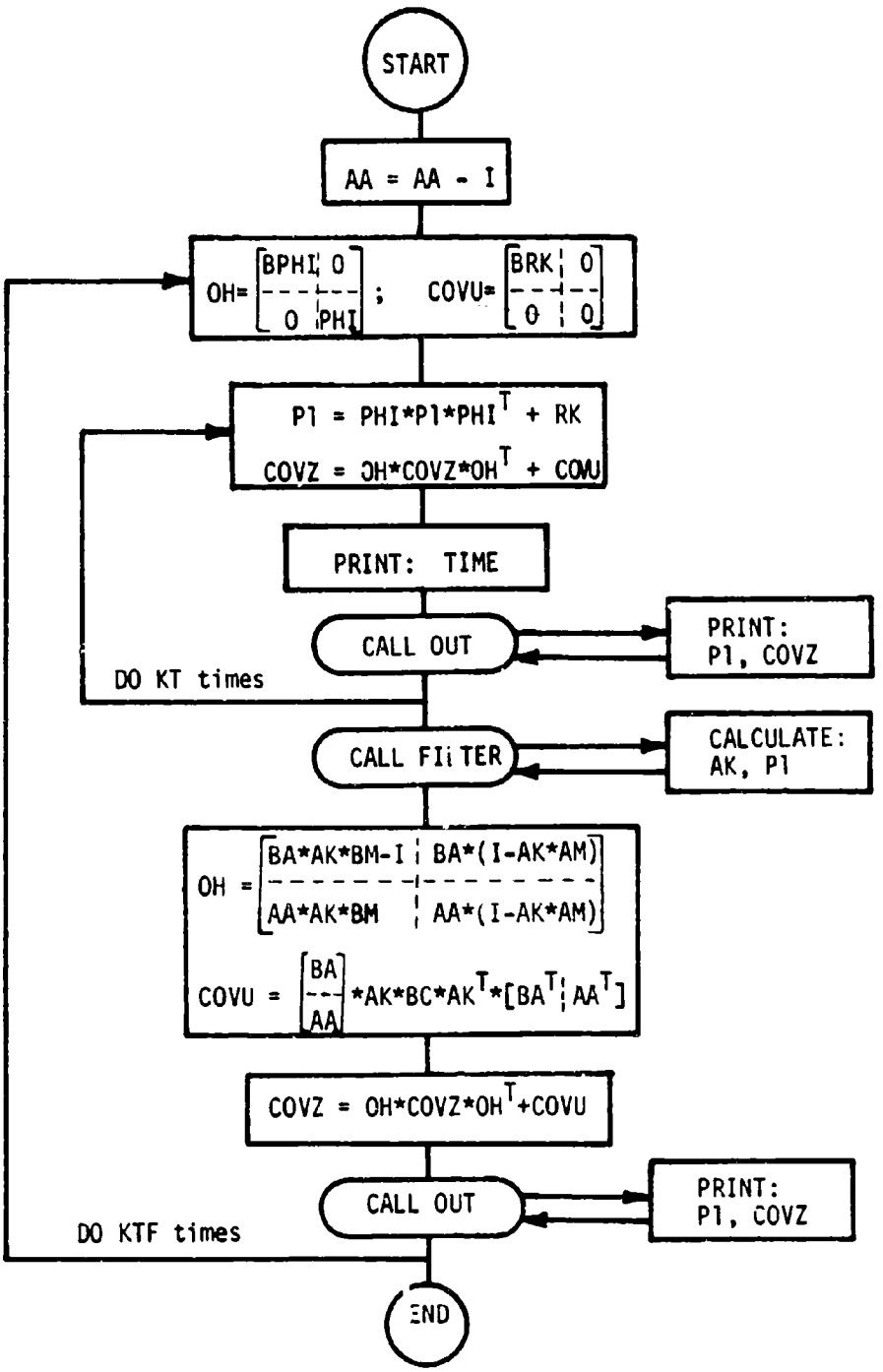


Fig. 9 Flow diagram of subroutine ERCOV.

denotes the number of measurements to be taken. Inside the DO-loop Θ_n and $Cov(U_n)$ are defined according to Eq. (2-34) and stored under the names OH and COVU respectively. Though these two matrices are constant, they are recalculated at each measurement due to the fact that they share the same storage locations as Θ_n' and $Cov(U_n')$. This way OH and COVU can be used for two purposes which saves useful storage space. Time-wise, this is justified as there are hardly any computations involved in calculating Θ_n and $Cov(U_n)$. The following DO-loop updates P1 and COVZ according to Eqs. (2-36) and (2-38). KT denotes the number of updates between measurements. Before each update the propagation time is printed out. P1 and COVZ are printed in subroutine OUT after each update. With the system updated KT times up to the measurement time T, a new measurement will be taken by transferring control to subroutine FILTER, where a new AK and P1 are calculated. In the statements following the return of control from FILTER, Θ_n' and $Cov(U_n')$ are calculated according to Eqs. (2-41) and (2-44) and are stored in OH and COVU. OH is obtained by first calculating the partitioned matrices, and augmenting these matrices in subroutine SUM. COVU is obtained by augmenting BA with AA - I, and post-multiplying and premultiplying $AK*BC*AK^T$ by the newly obtained matrix according to Eq. (2-44). As this augmented matrix contains only two matrices in this case, there is an entry SUMB provided in subroutine SUM which augments two matrices instead of four as is explained in the description of subroutine SUM. The new matrices P1 and COVZ are printed in subroutine OUT and the whole sequence of updating and calculating a new estimate is repeated again until the final measurement KTF is reached.

Subroutine FILTER: This subroutine of which the flow diagram is shown in Fig. 10 calculates the optimum gain AK and the new error in estimation P1. The parameter list in the subroutine and in the calling program is

FILTER(AM,AK,P1,AC,N,M)

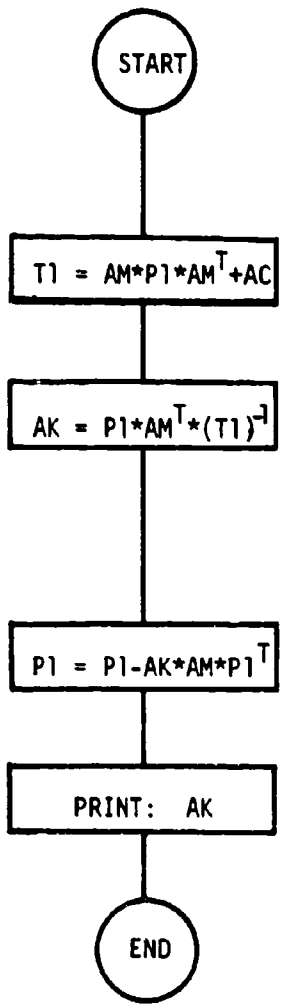
The equations used for the Kalman filter are strictly according to the equations derived in Chapter II, Eqs. (2-27) and (2-28). All variables except AK are to be defined in the calling program. P1 which is transferred to FILTER is equal to the error in estimation at the end of the last update when transferred back to the calling program P1 will be equal to the new corrected estimate according to Eq. (2-28). The function of the statements can easily be seen without further explanation.

Subroutine OUT: In this subroutine both P1 and COVZ are printed after each update and after each measurement. The parameter list in the subroutine and in the calling program is

OUT(P1,COVZ,T1,N,NB,NNB,IOUT)

T1 is a dummy array used for storing the square roots of the diagonal elements of P1 and COVZ. According to the flag IOUT either the full matrix of P1 and COVZ is printed when IOUT = 2, or the square root of just the diagonal terms is printed when IOUT = 1. With merely changing this small subroutine the print format can easily be changed without changing the binary deck of a large subroutine.

Subroutine DRUK: This subroutine is made to print a matrix and is called by



Calculate new optimum gain AK

Calculate new covariance matrix of the error in estimation, P1

Fig. 10 Flow diagram of subroutine FILTER.

DRUK(B,M,N)

where

B is the matrix to be printed

N the number of columns

M the number of rows.

The matrix A is printed row-wise, or in case the number of rows is larger than the number of columns, the matrix is automatically printed column-wise. The message "transpose" is then printed above the matrix. The print format used with the print statement is

```
2  FORMAT(1X,12E11.4)
   DO 3 I=1,N
3  PRINT 2, (B(I,J),J=1,M)
```

The print format or statements are easily changed to suit one's purpose.

Subroutine SUM and entry SUMB: In this subroutine an augmented matrix is obtained from four smaller matrices according to

$$S = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

The call statement is

SUM(A,B,C,D,S,I1,I2,J1,J2,I12,J12)

where

I1 is the number of rows in A and B

I2 is the number of rows in C and D

J1 is the number of columns in A and C

J2 is the number of columns in B and D

I12 is the number of rows in S

J12 is the number of columns in S

With the entry SUMB the following type of augmented matrix is obtained

$$S = \begin{bmatrix} A \\ \hline C \end{bmatrix}$$

The same parameter list is used for SUMB as for SUM, only B, D, and J2 do not have any significance.

Subroutine AOI: In this subroutine all elements of a matrix B are equated to zero when the flag KK = 1. Otherwise, the matrix B is equated to the identity matrix. The call statement is

AOI(B,N,M,KK)

where

N is the number of rows in B

M is the number of columns in B

KK is a flag.

Subroutine ABT: This subroutine performs the following matrix product $S = A*B^T$. The call statement is

ABT(A,B,S,K,L,M)

where

K is the number of rows in A and S

L is the number of columns in A and B

M is the number of rows in B and the number of columns in S.

Subroutine ABTAC: This subroutine performs either of the following two matrix operations

$S = A*B*A^T$ when the flag KK = 2

$S = A*B*C^T$ when the flag KK = 3.

The call statement is

ABTAC(A,B,C,T,S,K,L,M,N,KK)

where

T is a dummy array used to store the product of A and B

K is the number of rows in A, T, S

L is the number of rows in B and the number of columns in A

M is the number of columns in B, C, T

N is the number of rows in C and the number of columns in S.

In the case that KK = 2, the calling sequence is

ABTAC(A,B,A,T,S,K,L,L,K,2)

Advantage has been taken of the fact that S is symmetric in this case. Therefore only the upper-diagonal terms including the diagonal terms are calculated, and the sub-diagonal terms are equated to the corresponding elements above the diagonal. This way calculation time is saved.

The following matrix subroutines were implemented on the computer as library subroutines. Therefore, these subroutines are not contained in the listing and only the calling sequence is explained.

Subroutine MATINV: This subroutine is used to take the inverse of a matrix. Jordan's method is used to reduce a matrix A to the identity matrix I through a succession of transformations. With this method, the matrix equation A*X = B is solved, where A is a square coefficient matrix and B is a matrix of constant vectors. The inverse and the determinant of A are also computed. The calling sequence is

CALL MATINV(A,N,B,NB,DET,MA)

where

A is a square matrix of which the inverse has to be taken, as output it contains A^{-1}

N is the order of A

B is a two-dimensional array (not usually square), as output contains X

NB is the number of columns in B; if NB = 0, the routine is used only for the inversion

DET is the determinant of A calculated by MATINV

MA is the dimension of A in the calling program.

Subroutine MATSUB: In this subroutine two matrices are subtracted according to C = A - B. The calling sequence for this matrix routine is

```
Call MATSUB(A,B,C,N,NX,M)
```

where

N is the number of rows in A, B, and C

NX is the row dimension of the matrices in the calling program

M is the number of columns of A, B, and C

Subroutine MATADD: In this subroutine two matrices are added according to C = A + B. The calling sequence is given by

```
CALL MATADD(A,B,C,N,NX,M)
```

where

N is the number of rows in A, B, and C

NX is the row dimension of the matrices in the calling program

M is the number of columns in A, B, and C

Subroutine MATMPY: In this subroutine two matrices are multiplied according to C = A*B. The calling sequence is given by

```
CALL MATMPY(A,B,C,N,NX,M,MX,L)
```

where

N is the number of rows in A and C

NX is the row dimension of A and C in the calling program

M is the number of rows in B

MX is the row dimension of B in the calling program

L is the number of columns in B and C

CHAPTER IV

AN INERTIAL ESTIMATION PROBLEM

A. Errors Occurring in an Inertial Guidance System

In vehicles navigated by an inertial system it is desirable that the system computes the position and velocity with respect to earth very accurately. However, several types of errors occur in an inertial guidance system. These errors fall into two categories

1. deterministic
2. random

The deterministic errors are usually simple in form and quite easy to describe mathematically, such as errors with constant coefficients or with sinusoidal characteristics. These errors are generally compensated for, i.e., effectively subtracted out of the system. The random errors are treated statistically based upon a mathematical specification. Gyros, accelerometers, initial alignments, servos, digital or analog computers and geographical data are some examples of error sources that arise either within the inertial system or with outside data used by the inertial system. When dealing with errors in an error analysis it is necessary to describe these errors mathematically in order to study their propagation. Generally, these errors do not consist of pure white noise, but are correlated in time. This problem will be solved by adding states to the state vector x, and simulating the errors as being the outputs of stochastic processes with white noise inputs.

It is impossible to implement inertial guidance systems without

errors. These errors will grow very large after a period of time if they are not corrected. In order to keep the errors generated in an inertial system within acceptable bounds it is necessary to recalibrate the system periodically. The correction of the system errors is achieved by the use of independent sources of information. These external measurements can include position, velocity, attitude and combinations thereof. The external measurements are compared to corresponding quantities indicated by the inertial system. The Kalman filter uses the differences between indicated and measured quantities to provide the optimum estimate of the errors in the system. A block diagram of a recalibration scheme is contained in Fig. 11.

As well as estimating the states of the system, it is also important to obtain with the Kalman filter an estimate of those error sources which are correlated in time. If the error sources contain only white noise, estimating the errors would not assist in predicting the error at the next time of interest due to the fact that white noise is not correlated in time. However, when the disturbances and the measurement errors are not changing rapidly compared with the system state and measurements, the filter accuracy can be enhanced by estimating these errors. The estimation of the system disturbances and measurement errors which have significant correlation time, increases the number of state variables to be estimated. This is frequently described as "state vector augmentation."

In the navigation problem to be considered, three types of random variables are used

1. white noise

2. random constant

3. exponentially correlated random variable.

Of the three types of random signals, only the white noise is uncorrelated in time.

The white noise is denoted by w. The characteristics are: an expected value $E(w) = 0$, and an autocorrelation $R(\tau) = \sigma^2 \delta(\tau)$.

A random constant can be generated with the use of one additional state. This is illustrated in Fig. 12. The state differential equation can be written as

$$\dot{e} = 0$$

The initial condition is chosen according to the nature of the error. The autocorrelation is $\overline{e^2(0)}$.

The exponentially correlated random variable is frequently a useful representation of errors in inertial navigation systems. The autocorrelation function of the random signal is a declining exponential

$$E[e(t_1)\, e(t_2)] = \sigma^2\, e^{-\beta(t_2-t_1)} \qquad (4\text{-}2)$$

where σ^2 is the variance and β is the reciprocal of the correlation time. An exponentially correlated random variable can be generated by passing an uncorrelated signal, i.e., white noise, through a linear first-order feedback system. A block diagram for this stochastic process is shown in Fig. 13. The differential equation of the additional state variable is

$$\dot{e} = -\beta\, e + w \qquad (4\text{-}3)$$

where

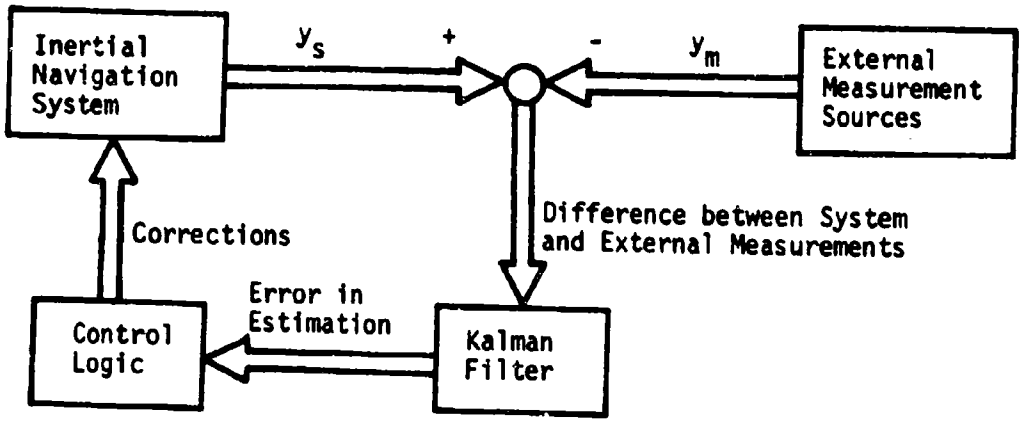


Fig. 11 Block diagram of a recalibration scheme.

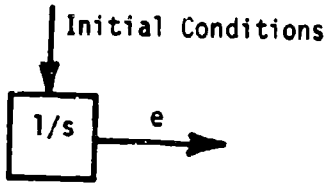


Fig. 12 Mathematical model for a random constant.

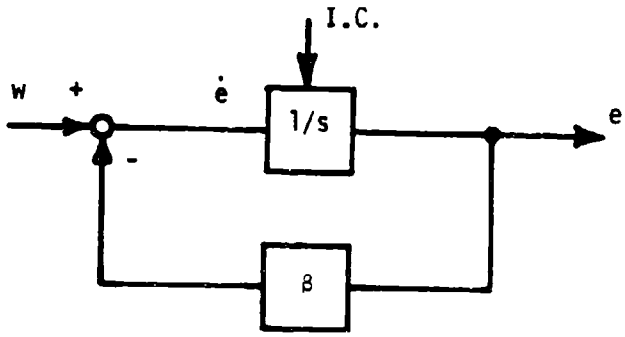


Fig. 13 Stochastic process with exponentially correlated output.

$$E(w) = 0,\ E[w(t)\ w(t+\tau)] = 2\ \beta\ \sigma^2\ \delta(\tau)$$

B. Simulation of an Inertial Guidance System

The model of the system contains seven states, not including the augmented states for the errors. The block diagram of the system is shown in Fig. 14, see [7]. This system is called a coupled model as all states are coupled resulting in a rather complicated F-matrix. For the optimum Kalman filter the same model is used in the filter as in the system. In order to obtain a less complicated Kalman filter, the equations are simplified, resulting in two other models of the system as shown in Figs. 15 and 16. The reference frame used is: x-north, y-east, z-down.

It is assumed that the accelerometer and gyro errors can be represented by an exponentially correlated random variable, Eq. (4-3). For this example only velocity measurements are considered, with measurement errors consisting of white noise with a random bias term, Eq. (4-1). The state vector used in all systems therefore is given by

$$x^T = [\delta R_x,\ \delta R_y,\ \delta \dot{R}_x,\ \delta \dot{R}_y,\ \phi_x,\ \phi_y,\ \phi_z,\ \nabla_x,\ \nabla_y,\ \epsilon_x,\ \epsilon_y,\ \epsilon_z,\ \nu_x,\ \nu_y] \quad (4\text{-}4)$$

where δR indicates position, ϕ the platform tilt, ∇ the position error, ϵ the tilt error, and ν the measurement error. The differential equations for the errors [8] are given by

$$\dot{\nabla} = -\beta_\nabla\ \nabla + w_\nabla \quad (4\text{-}5)$$

$$\dot{\epsilon} = -\beta_\epsilon\ \epsilon + w_\epsilon \quad (4\text{-}6)$$

$$\dot{\nu} = 0 \quad (4\text{-}7)$$

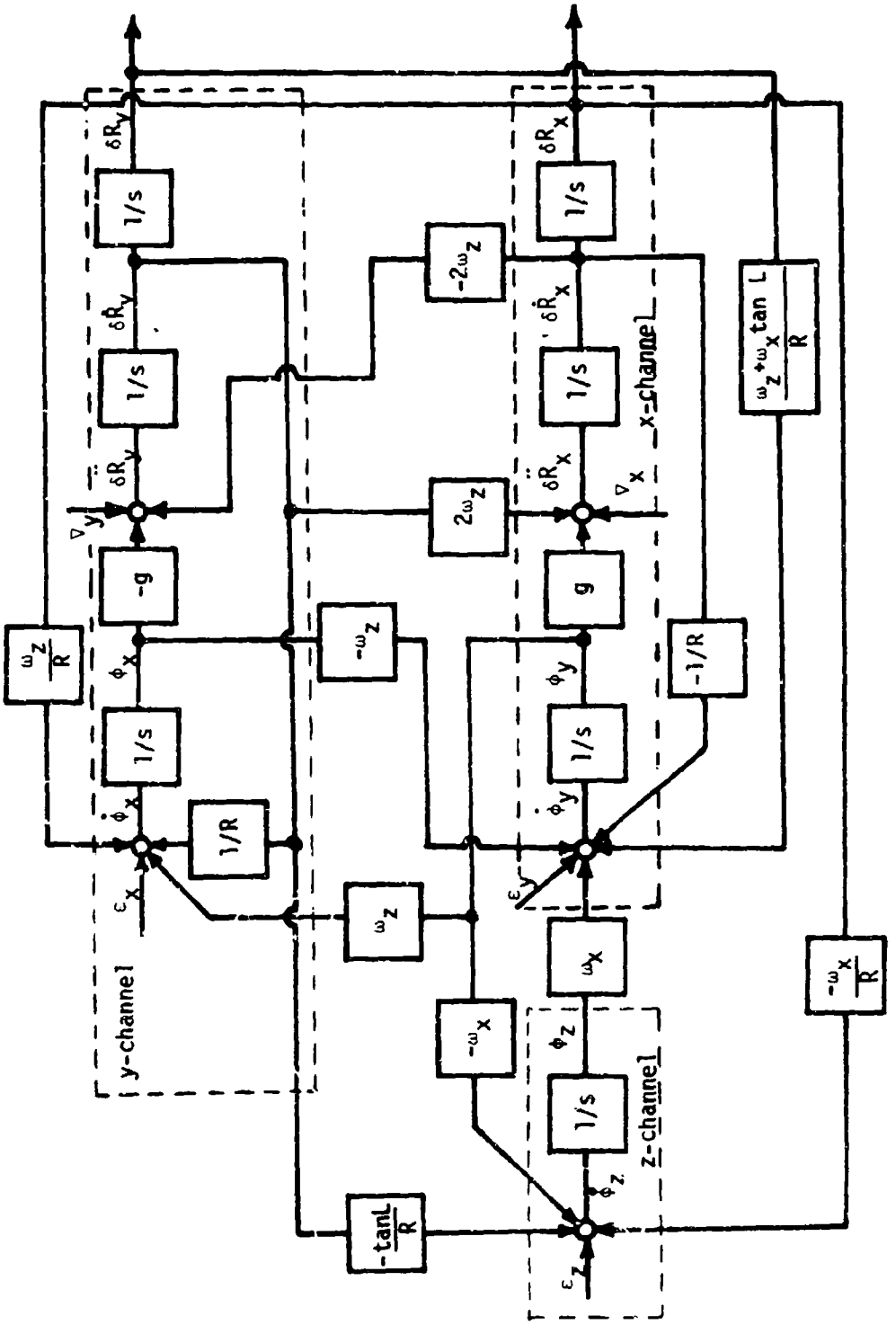


Fig. 14 Block diagram of an inertial navigation system, Model A.

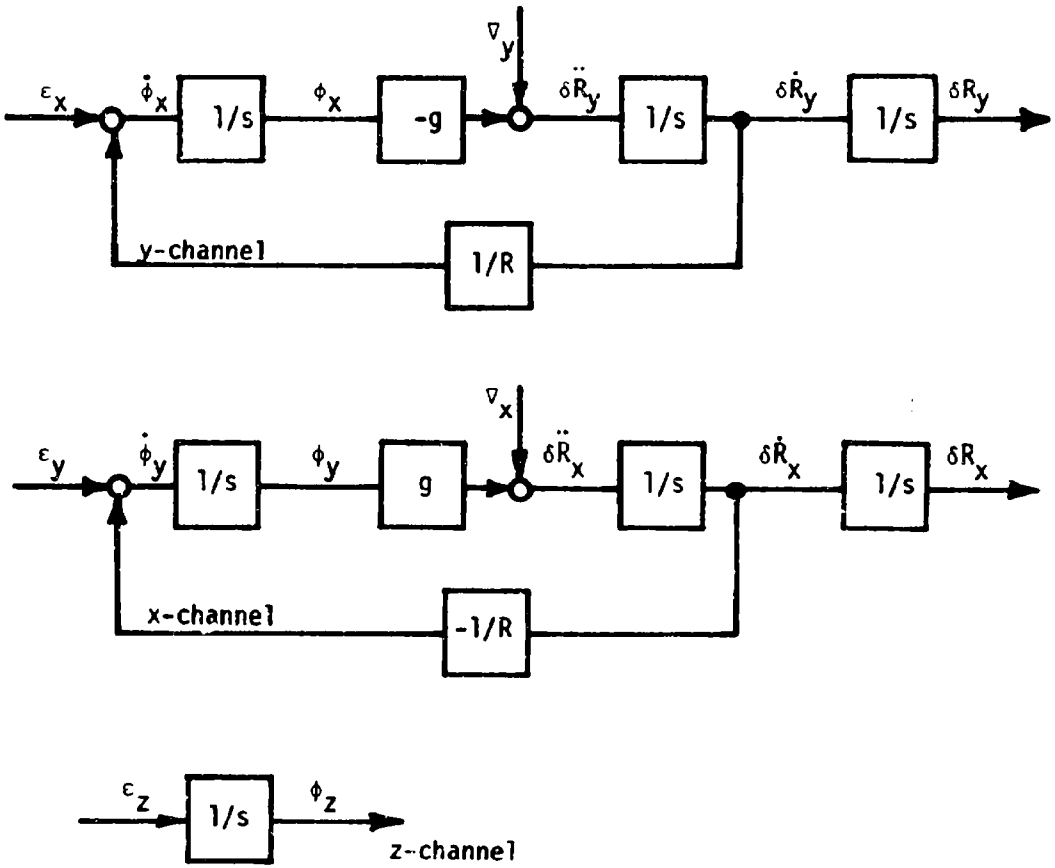


Fig. 15 Block diagram of an inertial navigation system, Model B.

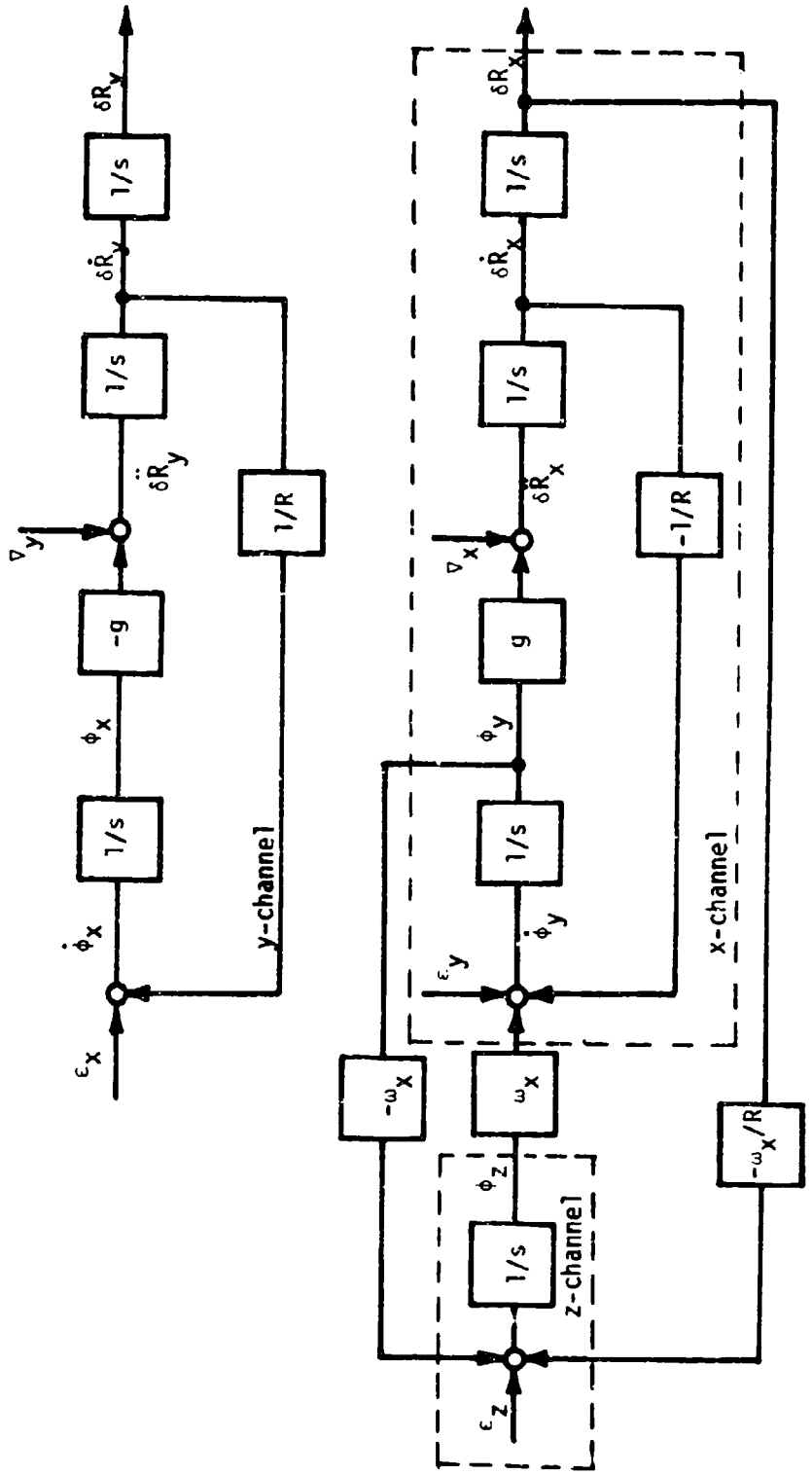


Fig. 16 Block diagram of an inertial navigation system, Model C.

For computational simplicity, the F-matrix is assumed to be non-varying with time, since the change in F(t) is negligible over the time interval studied. This means that the radius of the earth, R, and the gravity vector, g, are constants, and motion relative to the earth is neglected. The model shown in Fig. 14 is used for the dynamical simulation of the system. The F-matrix of this model with the augmented states for the simulation of the errors, is given by

$$F = \left[\begin{array}{ccccccc|ccccc|cc}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2\omega_z & 0 & g & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2\omega_z & 0 & -g & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\omega_z/R & 0 & 0 & 1/R & 0 & \omega_z & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & \psi & -1/R & 0 & -\omega_z & 0 & \omega_x & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-\omega_x/R & 0 & 0 & \frac{-\tan L}{R} & 0 & -\omega_x & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_{\nabla x} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_{\nabla y} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_{\varepsilon x} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_{\varepsilon y} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_{\varepsilon z} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right] \qquad (4\text{-}8)$$

where $\omega_x = \Omega \cos L$ and $\omega_z = -\Omega \sin L$, with Ω the angular rate of rotation of the earth; L is the latitude; and ψ represents the expression $\frac{\omega_z + \omega_x \tan L}{R}$. The system consists of three channels: the x-, y- and

z-channel, as shown by the dotted lines in Fig. 14. These channels are coupled by terms containing functions of the angular velocity, ω, and/or the latitude, L. These cross-coupling terms between the different channels of the system (Fig. 14) are generally much smaller than the other terms. This fact is used for a simplification of the Kalman filter equations. If all cross-coupling terms are ignored, the model in Fig. 15 is obtained. The upper left part of the new system matrix F becomes

$$F_{11} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & -g & 0 & 0 \\ 0 & 0 & 0 & 1/R & 0 & 0 & 0 \\ 0 & 0 & -1/R & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The order of the states in the state vector is changed to

$$x^T = [\delta R_y,\ \delta \dot{R}_y,\ \phi_x,\ \delta \dot{R}_x,\ \delta R_x,\ \phi_y,\ \phi_z] \tag{4-9}$$

A new F_{11}-matrix is obtained by rearranging the rows and columns accordingly.

$$F_{11} = \left[\begin{array}{ccc|ccc|c} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 1/R & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & -1/R & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{4-10}$$

The state vector matrix equation can now be decomposed into three independent sets of differential equations. If the states to simulate the errors are included, the three state vectors for these three independent systems are

$$x_1^T = [\delta R_y,\ \delta \dot{R}_y,\ \phi_x,\ \nabla_y,\ \varepsilon_x,\ \nu_y] \tag{4-11a}$$

$$x_2^T = [\delta R_x,\ \delta \dot{R}_x,\ \phi_y,\ \nabla_x,\ \varepsilon_y,\ \nu_x] \tag{4-11b}$$

$$x_3^T = [\phi_z,\ \varepsilon_z] \tag{4-11c}$$

The states to simulate the velocity measurement errors, ν_x and ν_y, are coupled to their corresponding channel through the measurement matrix. The F-matrices corresponding to the three state vectors are

$$F_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 1 & 0 & 0 \\ 0 & 1/R & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\beta_{\nabla y} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\beta_{\varepsilon x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4-12a}$$

$$F_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & g & 1 & 0 & 0 \\ 0 & -1/R & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\beta_{\nabla x} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\beta_{\varepsilon y} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4-12b}$$

$$F_3 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{4-12c}$$

The computer time and memory requirements are considerably reduced when the system in Fig. 14 is decomposed into a set of three independent systems. The equations for the Kalman filter can be programmed much more efficiently this way on a special-purpose computer.

A third model for the system is shown in Fig. 16. This model can be decomposed into a set of two independent systems. Without the augmented states to simulate the errors, the state vector and the corresponding system matrix F_{11} are given by

$$x^T = [\delta R_y,\ \delta \dot{R}_y,\ \phi_x,\ \delta R_x,\ \delta \dot{R}_x,\ \phi_y,\ \phi_z] \tag{4-13}$$

$$F_{11} = \left[\begin{array}{ccc|cccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 1/R & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & -1/R & 0 & \omega_x \\ 0 & 0 & 0 & -\omega_x/R & 0 & -\omega_x & 0 \end{array}\right] \tag{4-14}$$

The system matrix can be decomposed along the dotted lines. The states to simulate the random inputs can be added to the appropriate channel in analogy to Eq. (4-8) to obtain a full system matrix.

C. Analysis of Three Estimation Schemes

The models of the system in Figs. 14, 15 and 16 will be referred to as models A, B and C. Model A simulates the actual dynamics of the system very closely and will therefore be used to simulate the system. The following three estimation schemes have been used

1. Optimum Kalman, using model A for the simulation of the system and for the Kalman filter;
2. Sub-optimum Kalman, using model B for the filter; and
3. Sub-optimum Kalman, using model C for the filter.

The reference information for the estimation schemes is assumed to consist of velocity measurements only. The filter system matrix of model A is stored according to Eq. (4-8). The filter system matrices for model B and C are stored the same way, i.e., as a 14 x 14 matrix with the states in the same order as in Eq. (4-4). The change in the order of the states mentioned in the previous section is merely a way to illustrate how the F-matrix can be decomposed into smaller matrices in order to be programmed more efficiently on a special-purpose digital computer. For example, model A contains fourteen states, all of which are coupled, either through the system matrix or the measurement matrix, and therefore all the states have to be estimated simultaneously. However, model B contains only twelve states to be estimated. The two states in the z-channel, ϕ_z and ϵ_z, are not affected by the optimum gain as they are uncoupled from the observable states, $\delta\dot{R}_x$ and $\delta\dot{R}_y$. The resulting twelve-state system can be decomposed into two uncoupled systems each with six states, of which the system matrices are given by Eqs. (4-12a) and (4-12b). The optimum gain and the estimate can then be calculated for each of these systems separately. This results in storing more matrices, but of smaller size, than with the system composed of all the states as would be done for model A. Model C contains fourteen states to be estimated since the z-channel is coupled with the x-channel. The x- and y-channels are uncoupled, so that the system can

be decomposed into two systems, one with six states for the y-channel, and the other with eight states to simulate the x- and z-channels. Here also the new estimate and the optimum gain can be calculated separately for each of the two systems.

The external measurements used are the velocities in the x- and y-directions, $\delta\dot{R}_x$ and $\delta\dot{R}_y$. Realizing that the measurement errors have to be added to the output also, Eq. (2-1b) becomes

$$y = \begin{bmatrix} \delta\dot{R}_x + \nu_x \\ \delta\dot{R}_y + \nu_y \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{4-15}$$

where v_x and v_y are the "white noise" components of the measurement errors and ν_x and ν_y are the bias terms of the measurement errors. The measurement matrix M for the system will be

$$M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-16}$$

The output vector for the filter in Eq. (2-23) is likewise obtained and results in the same measurement matrix as for the system in Eq. (4-16).

Table I contains the values for the different parameters used and their values when converted into the units of feet, seconds and radians.

TABLE I System Parameters

g	R	Ω	L	β_ε	β_∇
		15°/hr	45°	1/4 hrs^{-1}	1/2 hrs^{-1}
32.2 ft/sec^2	2.1×10^7 ft	$.944\times10^{-4}$ rad/sec	.785 rad	6.95×10^{-5} sec^{-1}	.139 sec^{-1}

$\omega_x = \Omega \cos L = .666\times10^{-4}$ sec^{-1}

$\omega_z = -\Omega \sin L = -.666\times10^{-4}$ sec^{-1}

The initial conditions for the error in estimation, x, are contained in Table II in addition to their values in the units of feet, seconds and radians.

TABLE II Initial Conditions

$R_{x,y}$	$\dot{R}_{x,y}$	$\phi_{x,y}$	ϕ_z	$\sigma_{\nabla x,y}$	$\sigma_{\varepsilon x,y,z}$	$\sigma_{vx,y}$
3000 ft	1. ft/sec	10 $\widehat{\text{min}}$	10°	20 $\widehat{\text{sec}}$	.01°/hr	.3 ft/sec
$3.\times10^{3}$ ft	1. ft/sec	2.916×10^{-3} rad	.175 rad	3.12×10^{-3} rad ft/sec^2	4.925×10^{-8} rad/sec	.3 ft/sec

The initial condition for the covariance matrix of the error in estimation P(0) is obtained by squaring the values in Table II. The covariance of the random driving terms is

$$\text{Cov}(w_\nabla) = 2\beta_\nabla\sigma_\nabla^2 = 2.71\times10^{-9} \text{ rad ft}^2/\text{sec}^4$$

$$\text{Cov}(w_\varepsilon) = 2\beta_\varepsilon\sigma_\varepsilon^2 = 3.37\times10^{-19} \text{ rad/sec}^3$$

The values for β_∇, β_ε, σ_∇ and σ_ε are obtained from Tables I and II. The diagonal of both the covariance matrix for the system and the filter is obtained from the above values for the random driving terms, and all the off-diagonal terms are zero as these random signals are uncorrelated.

The covariance of the "white noise" error components of the measurements is given by

$$\text{Cov}(v_x) = \text{Cov}(v_y) = 1/4 \text{ ft}^2/\text{sec}^2$$

The covariance matrix of these errors is given by

$$C = \begin{bmatrix} .25 & 0 \\ 0 & .25 \end{bmatrix}$$

For the control matrices A_1 and A_2 the identity matrix is used. The initial conditions for Cov(z) can, with Eq. (4-45) and Eq. (4-46), be represented by

$$\mathrm{Cov}(z_0) = \begin{bmatrix} P_0 & 0 \\ 0 & 0 \end{bmatrix}$$

The first half of the diagonal terms are equal to the diagonal elements of P(0). The terms in the second half of the diagonal and all off-diagonal terms are equal to zero.

For the simulations, a measurement time T = 60 sec was used. The number of updates KT = 2. The number of measurements KTF = 30, corresponding to a final time of t = 1/2 hour.

D. Results and Comments

The results of the simulations with the different models are plotted in Figs. 17 through 20. The figures contain the estimation errors in δR_x, $\delta \dot{R}_x$, ϕ_y and ϕ_z for all three models. The y-channel results were not plotted since they are for all three models very much like the results for the x-channel of the optimum case. One can notice that model A gives the best estimate, closely followed by the results with model C. The estimate obtained with model B is far worse. However, it is misleading in this case to compare the three estimation schemes when the measurement time is the same for all three cases. The calculation of the optimum gain takes much longer with the complicated model A than with the far less complicated model B. Therefore with models B and C more measurements can be taken and more estimates calculated in the same period of time than with model A. The more measurements taken, the

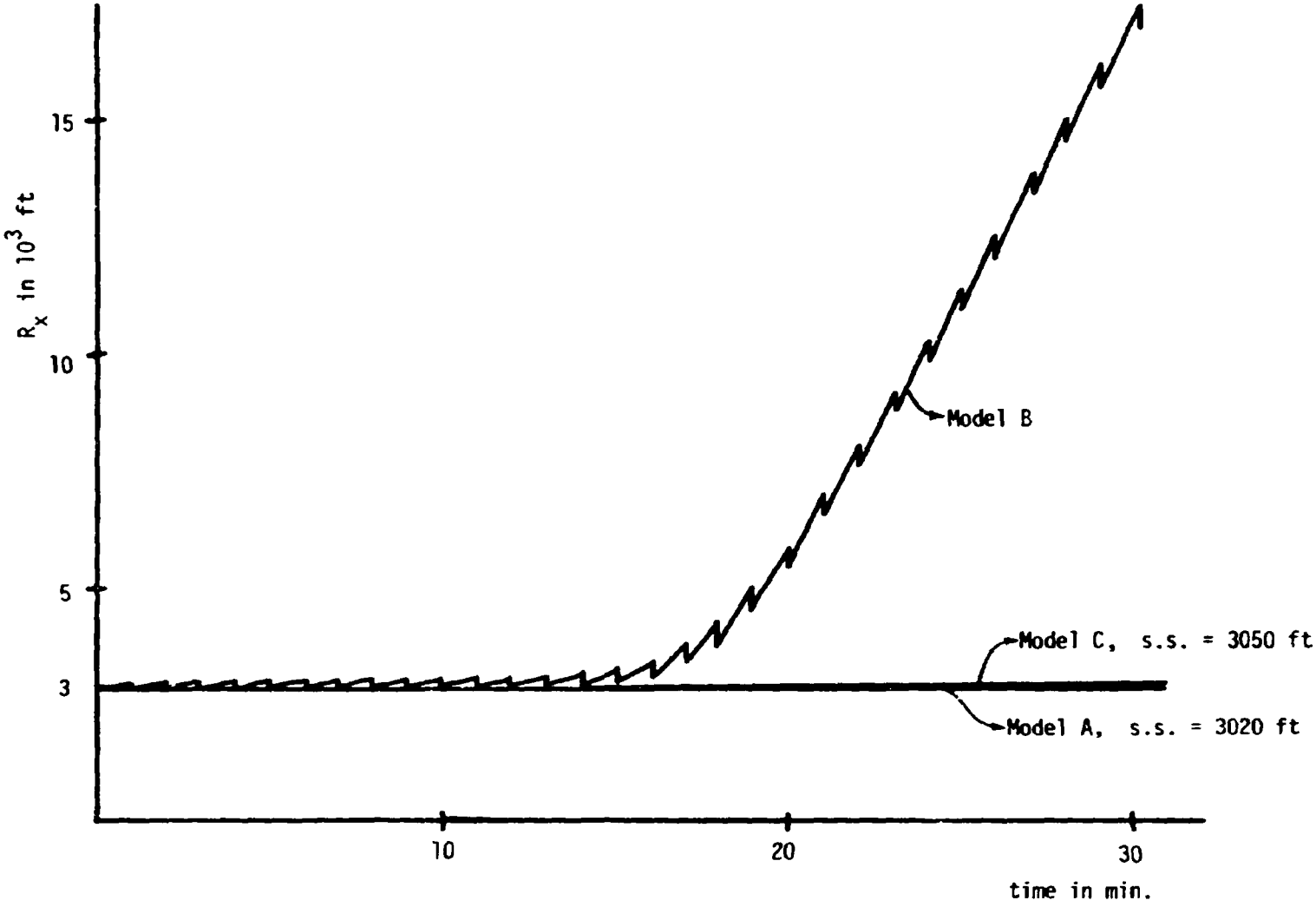


Fig. 17 Error in estimation of the position in the x-direction.

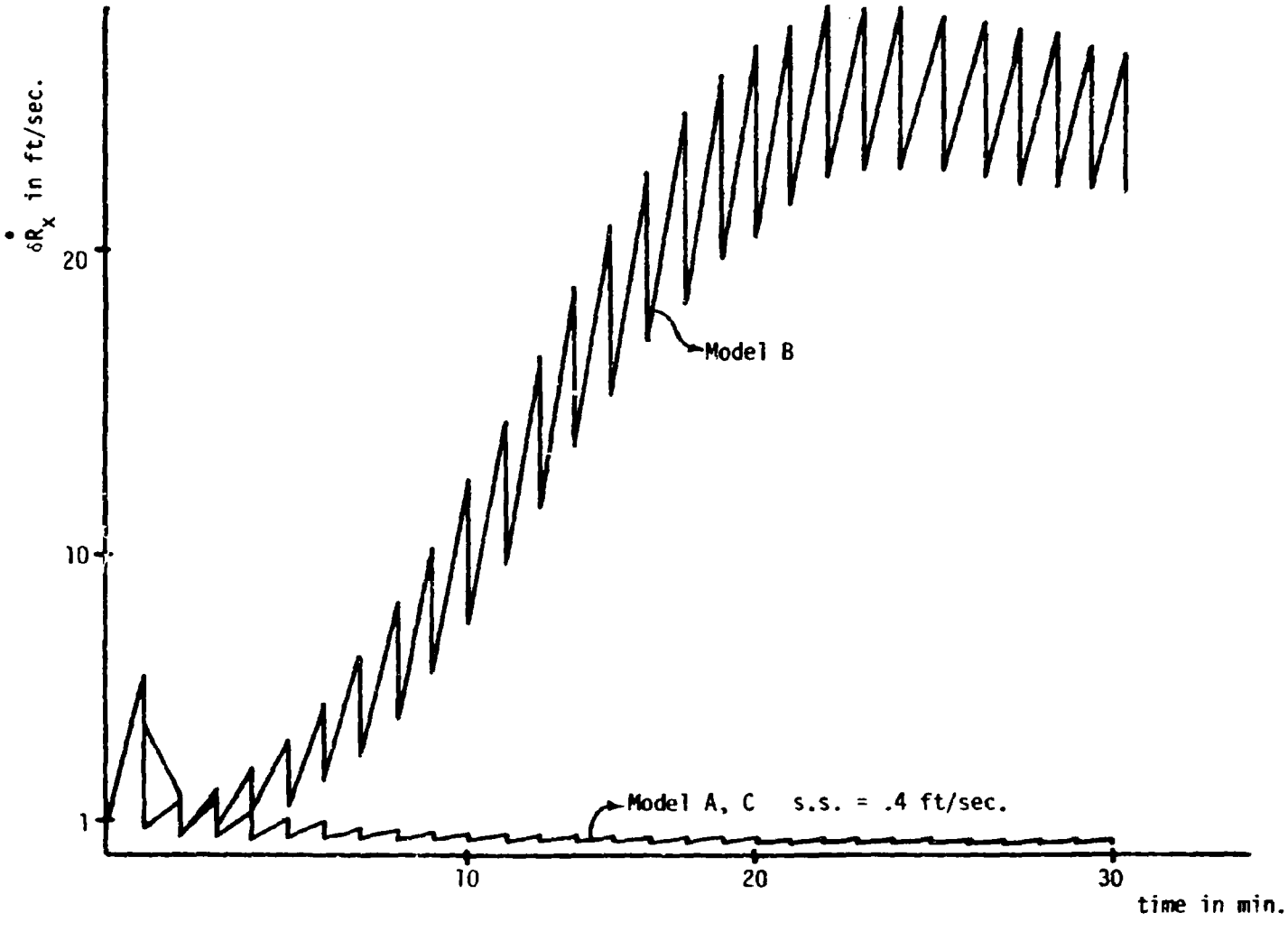


Fig. 18 Error in estimation of the velocity in the x-direction.

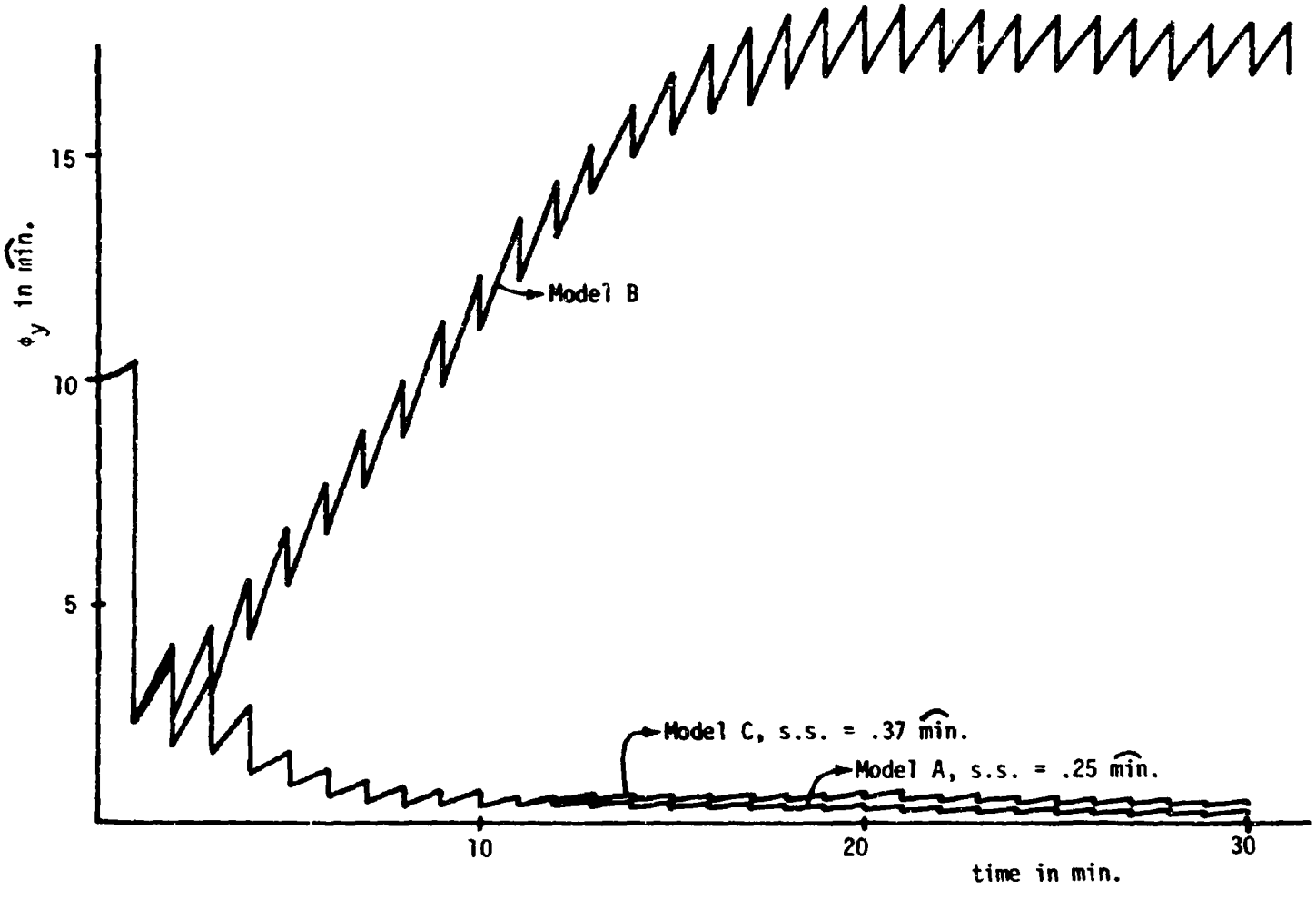


Fig. 19 Error in estimation of the platform tilt.

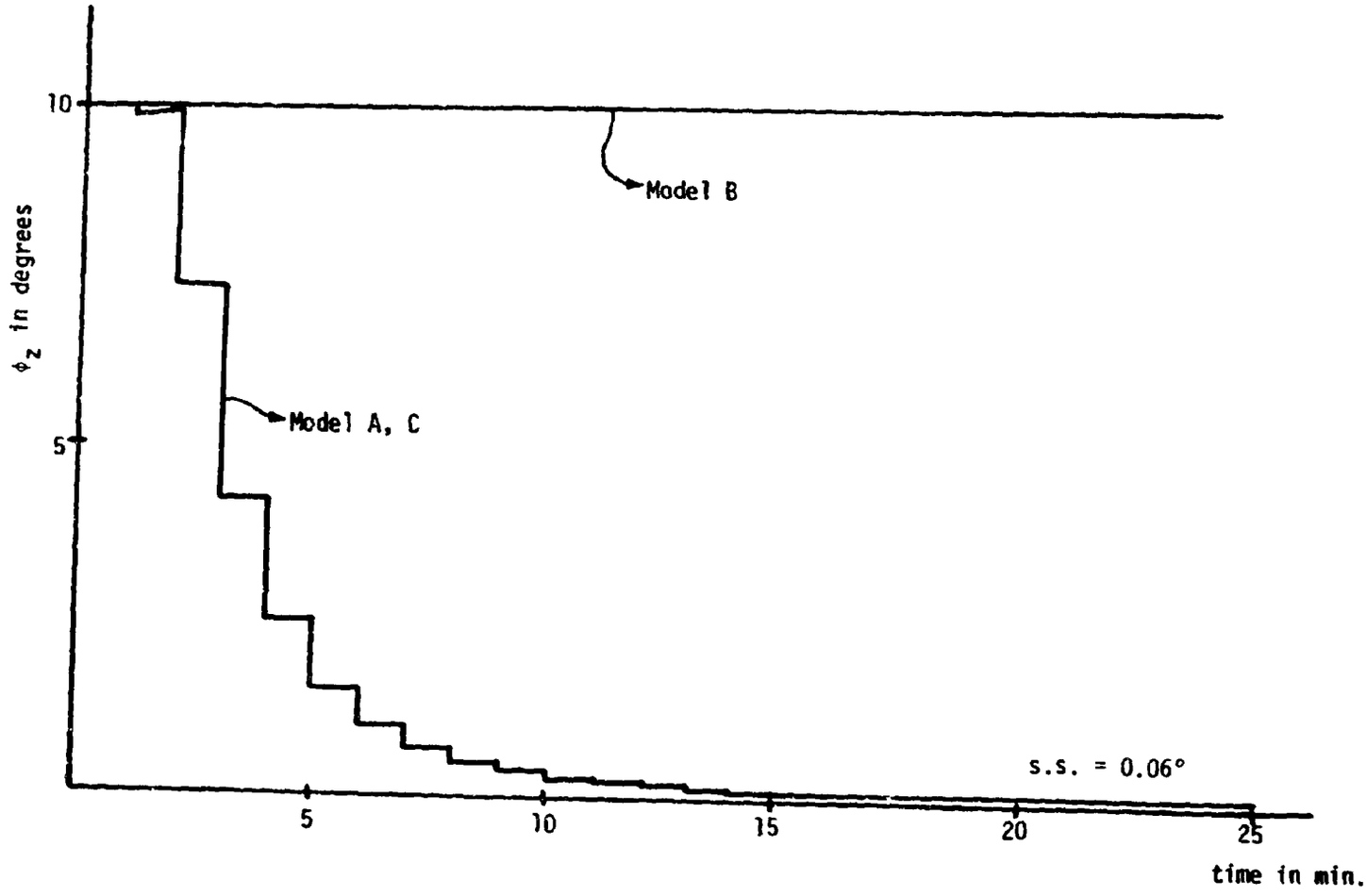


Fig. 20 Error in estimation of the azimuth.

faster the steady state is reached, and due to the small correlation time of the errors, the steady state may be smaller.

In order to get a better idea about the trade-offs, Table III contains for each of the three models:

1. The amount of storage necessary for the matrices;
2. The time involved to calculate a new estimate; and
3. The steady state for the states of the system after 30 minutes.

The amount of storage given in Table III does not include the memory necessary for the instructions. The matrices include: AF, AQ, PHI, RK, AM, AC, P1, AK, T1 and T2.

TABLE III Summary of the Results

	Model A	Model B	Model C
Memory necessary to store all matrices (words)	1432	484	728
Time of calculations	1.	.16	.26
Steady state after 30 minutes			
δR_x in 10^3 ft	3.02	17.5*	3.05
δR_y in 10^3 ft	3.05	3.10	3.05
$\delta \dot{R}_x$ in ft/sec	.394	21.4	.395
$\delta \dot{R}_y$ in ft/sec	.373	.811	.374
ϕ_x in $\widehat{min}$	.192	.407	.338
ϕ_y in $\widehat{min}$	.245	16.6	.362
ϕ_z in degrees	.058	9.9	.060

*This state does not reach steady state, but will continue to grow (see Fig. 17) due to velocity error.

In Chapter IV, Section C it was already shown that by ignoring the cross-coupling terms of the system the system could be decomposed into two different uncoupled models of the system, resulting in a smaller storage space required for the various matrices. Besides gaining a considerable amount of storage space, especially with large systems, there is also a great amount of computer time gained. Most of the calculations consist of multiplications of matrices. To multiply two n x n matrices, there are n^3 multiplications needed. If the dimension of the matrices is reduced by a factor 2, the amount of calculation is reduced by a factor $2^3 = 8$.

Besides the fact that some matrices are reduced in size by decomposing the system matrix, the transition matrix, PHI, and the covariance matrix of the error term for the random input of the filter, RK, usually converge more rapidly. This is an important factor when these two matrices have to be re-calculated at every time a new estimate is calculated. This is the case when the measurement time is not constant, and/or when the system matrix in the filter is time-varying. It should be noted here that the convergence of PHI and RK depends highly on the arrangement of the rows and columns. The order of the states in the state vector should be arranged in such a way that all elements of the system matrix are as close as possible to the diagonal in order to obtain rapid convergence.

The calculation of the time involved to calculate a new estimate is derived by assuming that most of the computer time is taken up by multiplying matrices — in particular, the matrices with the largest dimensions. These are: AF, AQ, PHI, RK, P1, T1 and T2. By

decomposing the system matrix, AF, all these matrices are reduced in size in the same way as AF. With model A, the multiplication of two matrices will involve $14^3 = 2744$ calculations. The system matrix of model B is split into two 6 x 6 matrices, therefore the equivalent multiplication takes $2 \times 6^3 = 432$ calculations. The system matrix of model C is split into an 8 x 8 matrix and a 6 x 6 matrix. The equivalent multiplication here involves $8^3 + 6^3 = 728$ calculations. It is assumed that the number of these calculations is proportional to the time of calculating a new estimate. In Table III, the time to calculate a new estimate for model A is taken to be 1.0, and all other times are relative to this model.

Apparently an estimator using model C in the Kalman filter is the most desirable, as it can be programmed quite efficiently by decomposing it into two uncoupled systems, and still the system reaches steady state in a short time. The reason that model B exhibits a strong growth of the errors in the x-channel is because the azimuth, ϕ_z, has been uncoupled in this model which makes ϕ_z unobservable. Therefore ϕ_z is unaffected by the optimum gain. Model B may be better suited for use when more states are observable.

APPENDIX A

The Matrix Exponential Equation

An important subroutine in the program is the matrix exponential (MEXP) subroutine. This routine utilizes the system matrix F to calculate the transition matrix which is defined by

$$\Phi(\Delta t) = e^{F\Delta t} \tag{A-1}$$

To evaluate this equation, Φ is put in a series form which can be written as

$$\Phi(\Delta t) = I + F\,\Delta t + F^2 \frac{\Delta t^2}{2!} + F^3 \frac{\Delta t^3}{3!} + \cdots \tag{A-2}$$

This series converges [9] and can easily be programmed on a digital computer. The flow diagram of this subroutine is shown in Fig. (A-1).

The Error Covariance Equation

From the system equation

$$\dot{x}(t) = F\,x(t) + w(t)$$

where $w(t)$ is a random noise process, a relation has to be obtained describing $x(t)$ at any instant of time as a function of an initial condition $x(0)$. Due to the random noise, a statistical description of the system is necessary. The covariance matrix of the states is given by

$$P(t) = E[x(t)\,x^T(t)]$$

with $E[x(t)] = 0$ if $E[x(0)] = 0$.

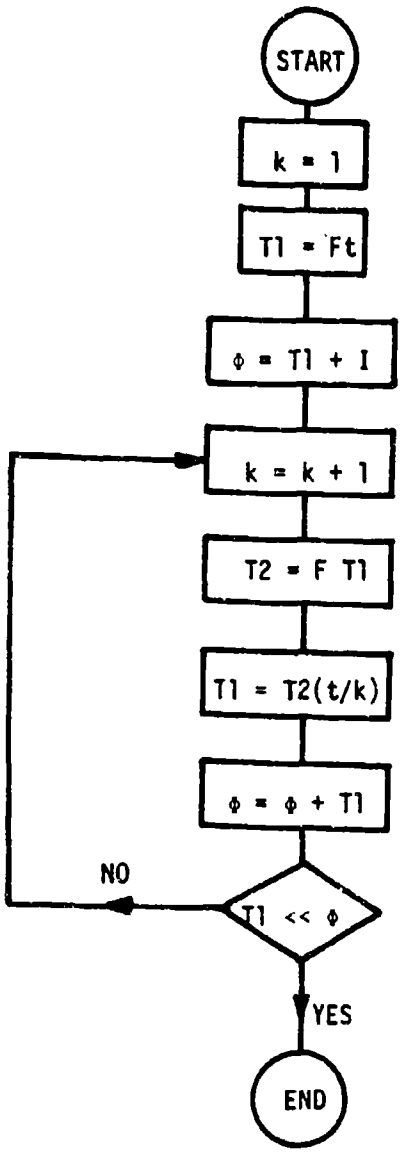


Fig. (A-1) Flow diagram for the numerical solution of the matrix exponential equation.

To generate the covariance matrix P(t) as a function of time and an initial condition, its differential equation is to be examined as follows

$$\dot{P}(t) = E[\dot{x}(t)\ x^T(t)] + E[x(t)\ \dot{x}^T(t)] \qquad \text{(A-3)}$$

Substituting the system equation yields

$$E[\dot{x}(t)\ x^T(t)] = E[(F\ x(t) + w(t))\ x^T(t)]$$
$$= F\ P(t) + E[w(t)\ x^T(t)]$$

Likewise,

$$E[x(t)\ \dot{x}^T(t)] = P(t)\ F^T + E[x(t)\ w^T(t)] \qquad \text{(A-4)}$$

Noting that x(t) can be written as

$$x(t) = \phi(t_o,\ t)\ x(t_o) + \int_{t_o}^{t} \phi(\tau,\ t)\ w(\tau)\ d\tau$$

and substituting into Eq. (A-4) yields

$$E[x(t)\ w^T(t)] = E[\{\phi(t_o,\ t)\ x(t_o)\}\ w^T(t)] +$$
$$+ E[\{\int_{t_o}^{t} \phi(\tau,\ t)\ w(\tau)\ d\tau\}\ w^T(t)]$$
$$= E[\int_{t_o}^{t} \phi(\tau,\ t)\ w(\tau)\ w^T(t)\ d\tau]$$
$$= \int_{t_o}^{t} \phi(\tau,\ t)\ E[w(\tau)\ w^T(t)]\ d\tau \qquad \text{(A-5)}$$

According to Eq. (2-5)

$$E[w(\tau)\ w^T(t)] = Q\ \delta(t - \tau)$$

Substituting into Eq. (A-5) yields

$$E[x(t)\ w^T(t)] = \int_{t_o}^{t} \phi(\tau,\ t)\ Q\ \delta(t - \tau)\ d\tau \qquad \text{(A-6)}$$

Fig. (A-2) shows the impulse function $\delta(t - \tau)$.

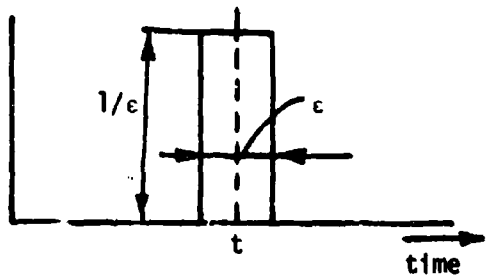


Fig. (A-2) Impulse function $\delta(t - \tau)$.

With a pulse width ε, the impulse function is only non-zero between $t - 1/2\ \varepsilon$ and $t + 1/2\ \varepsilon$. Therefore, the lower integration limit in Eq. (A-6) can be changed from t_0 to $t - 1/2\ \varepsilon$ without changing the result. Because ε is infinitely small,

$$\Phi(\tau, t) = \Phi(t, t) = I$$

As the impulse is only integrated over half its area, from $t - 1/2\ \varepsilon$ to t, the final result of Eq. (A-5) contains the factor 1/2.

$$E[x(t)\ w^T(t)] = 1/2\ Q$$

Substituting this result into Eq. (A-3) yields a simplified matrix Ricatti equation

$$\dot{P}(t) = P(t)\ F^T + F\ P(t) + Q \qquad \text{(A-7)}$$

The solution to this differential equation is obtained by the following method. By direct substitution it can be shown that

$$P(t) = [\Theta_{21} + \Theta_{22}\ P(0)][\Theta_{11} + \Theta_{12}\ P(0)]^{-1} \qquad \text{(A-8)}$$

where

$$\dot{\Theta}(t) = z\ \Theta(t) \qquad \text{(A-9)}$$

$$\Theta(0) = \left[\begin{array}{c|c} \Theta_{11}(0) & \Theta_{12}(0) \\ \hline \Theta_{21}(0) & \Theta_{22}(0) \end{array}\right] = I \qquad \text{(A-10)}$$

$$z = \left[\begin{array}{c|c} -F^T & 0 \\ \hline Q & F \end{array}\right] \qquad \text{(A-11)}$$

Proof: Rearranging Eq. (A-8) and taking the time derivative yields:

$$P(t)[\Theta_{11} + \Theta_{12}\, P(0)] = [\Theta_{21} + \Theta_{22}\, P(0)]$$

$$\dot{P}(t)[\Theta_{11} + \Theta_{12}\, P(0)] + P(t)[\dot{\Theta}_{11} + \dot{\Theta}_{12}\, P(0)] = [\dot{\Theta}_{21} + \dot{\Theta}_{22}\, P(0)] \qquad \text{(A-12)}$$

From Eq. (A-9)

$$\begin{aligned} \dot{\Theta}_{11} &= z_{11}\, \Theta_{11} \\ \dot{\Theta}_{12} &= 0 \\ \dot{\Theta}_{21} &= z_{21}\, \Theta_{11} + z_{22}\, \Theta_{21} \\ \dot{\Theta}_{22} &= z_{22}\, \Theta_{22} \end{aligned} \qquad \text{(A-13)}$$

where $z_{12} = 0$ according to Eq. (A-11), and $\Theta_{12} = 0$ which becomes apparent later on in Eq. (A-16) when the expression for $\Theta(t)$ is found. Substituting these results into Eq. (A-12) yields

$$\dot{P}(t) = -P(t)\, z_{11} + z_{21} + z_{22}[\Theta_{21} + \Theta_{22}\, P(0)]\, \Theta_{11}^{-1}$$

Inserting the values for z from Eq. (A-11)

$$\dot{P}(t) = P(t)\, F^T + Q + F\, P(t) \qquad \text{(A-14)}$$

which proves that both equations are equivalent.

In order to find an algorithm to program Eq. (A-8), the series method is used. From Eqs. (A-9) and (A-11)

$$\Theta(t) = e^{\left[\begin{array}{c|c} -F^T & 0 \\ \hline Q & F \end{array}\right] t} \tag{A-15}$$

Using the matrix exponential method to solve for $\Theta(t)$ yields

$$\Theta(t) = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right] + \left[\begin{array}{c|c} -F^T & 0 \\ \hline Q & F \end{array}\right] t + \left[\begin{array}{c|c} -F^T & 0 \\ \hline Q & F \end{array}\right]^2 \frac{t^2}{2!} + \left[\begin{array}{c|c} -F^T & 0 \\ \hline Q & F \end{array}\right]^3 \frac{t^3}{3!}$$

Performing the matrix multiplications yields

$$\Theta(t) = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right] + \left[\begin{array}{c|c} -F^T & 0 \\ \hline Q & F \end{array}\right] t + \left[\begin{array}{c|c} (-F^T)^2 & 0 \\ \hline -QF^T+FQ^T & F^2 \end{array}\right] \frac{t^2}{2!} +$$

$$+ \left[\begin{array}{c|c} (-F^T)^3 & 0 \\ \hline F(FQ^T-QF^T)-(FQ^T-QF^T)F^T & F^3 \end{array}\right] \frac{t^3}{3!} + \cdots \tag{A-16}$$

By noting how the series progresses, one can easily obtain simple expressions for Θ_{11}, Θ_{12} and Θ_{22} as follows

$$\Theta_{11}(t) = e^{-F^T t} = (\Phi^{-1})^T$$

$$\Theta_{22}(t) = e^{Ft} = \Phi$$

$$\Theta_{12}(t) = 0$$

With these results, Eq. (A-8) can be written in a simpler form

$$P(t) = \Phi\, P(0)\, \Phi^T + \Theta_{21}\, \Theta_{11}^{-1} \tag{A-17}$$

Θ_{21} can be easily found by using the matrix exponential method for solving $\Theta = e^{Zt}$. This method, however, takes a substantial amount of time on the computer due to the large dimension of z; therefore another method will be used. In the following derivation it turns out that for $\Theta_{12}\, \Theta_{11}^{-1}$ a rather simple-to-program expression can be found. Multiplying

the series of Θ_{21} and Θ_{11}^{-1} gives the result

$$\Theta_{21}\,\Theta_{11}^{-1} = Qt + (QF^T + FQ^T)\,\frac{t^2}{2!} + [(QF^T + F^TQ)F^T + F(QF^T + FQ^T)^T]\,\frac{t^3}{3!} + \cdots$$

This equation can be written in another form, realizing that Q is symmetric, or $Q = Q^T$, which makes $QF^T = (FQ)^T$, so that $FQ^T + QF^T$ is again symmetric.

$$\Theta_{21}\,\Theta_{11}^{-1} = Qt + (FQ + (FQ)^T)\,\frac{t^2}{2} + [F(FQ + (FQ)^T) + \{F(FQ + (FQ)^T)\}^T]\,\frac{t^3}{3!} + \cdots \qquad \text{(A-18)}$$

This series is easy to program with a simple iteration routine shown in Fig. (A-3), with $\Theta_{21}\,\Theta_{11}^{-1} = R_n$.

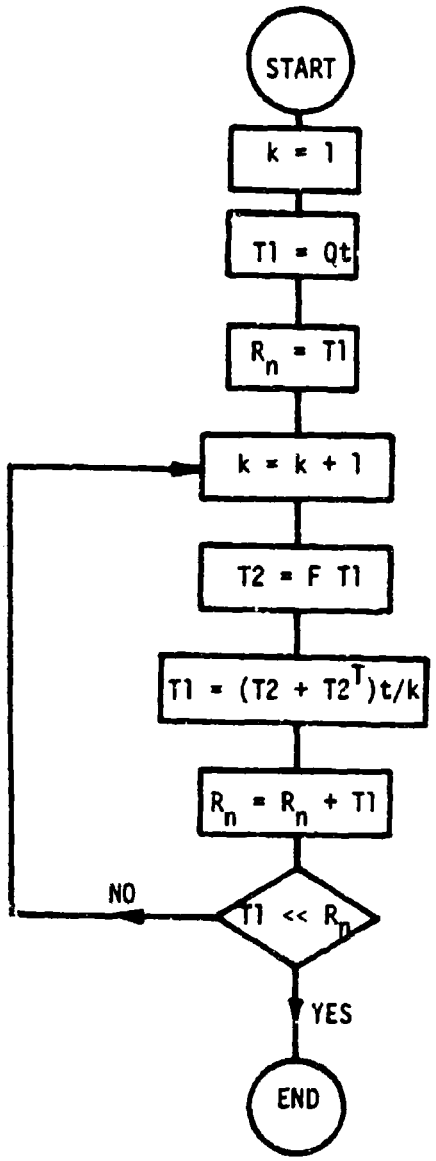


Fig. (A-3) Flow diagram for the numerical solution of the matrix Ricatti equation.

APPENDIX B

This appendix contains a listing of the sub-optimum Kalman filter estimation program explained in Chapter III.

```
      PROGRAM  ESTIM
      DIMENSION AF(20,20),BF(25,25),AQ(20,20),BQ(25,25),PHI(20,20),
     1BPHI(25,25),RK(20,20),BRK(25,25),AM(10,20),BM(10,25),AK(20,10),
     2AC(10,10),P1(20,20),BC(10,10),COVU(45,45),COVZ(45,45),AA(20,20),
     3BA(25,20),TA(25,25),TC(25,25),OH(45,45)
      COMMON T1(50,25),T2(50,25),T3(50,25),T4(25,25),T5(25,25)
      EQUIVALENCE (AF,BPHI),(AQ,BRK),(BF,T4),(BQ,T5),(T1,TA),(T2,TC)
    1 FORMAT(8I5,3E10.5)
    3 FORMAT(/*  PHI*)
    4 FORMAT(/*  BPHI*)
    5 FORMAT(/*  RK*)
    6 FORMAT(/*  BRK*)
    7 FORMAT(*1   N    NB    M    KT   KTF   KMAX  IOUT   KAI      T      AE      RE*)
      DO 222 JJ=1,10
      PRINT 7
      READ 1,N,NB,M,KT,KTF,KMAX,IOUT,KAI,T,AE,RE
      IF(EOF,60) 223,28
   28 PRINT 1,N,NB,M,KT,KTF,KMAX,IOUT,KAI,T,AE,RE
      D=T/FLOAT(KT)
      NNB=N+NB
      CALL AINPUT(AF,BF,AQ,BQ,AM,BM,AC,BC,P1,COVZ,AA,BA,N,NB,M,NNB,KAI)
      PRINT 3
      CALL MEXP(D,AF,AF,T1,T2,T3,PHI,N,KMAX,AE,RE, 1)
      PRINT 5
      CALL MEXP(D,AF,AQ,T1,T2,T3,RK,N,KMAX,AE,RE, 2)
      PRINT 4
      CALL MEXP(D,BF,BF,T1,T2,T3,BPHI,NB,KMAX,AE,RE, 1)
      PRINT 6
      CALL MEXP(D,BF,BQ,T1,T2,T3,BRK,NB,KMAX,AE,RE, 2)
      CALL ERCOV(PHI,BPHI,RK,BRK,AM,BM,AK,AC,BC,P1,OH,COVU,COVZ,AA,BA,TA
     1,TC,N,NB,M,NNB,KT,KTF,IOUT,D)
  222 CONTINUE
  223 CONTINUE
      END
```

```
      SUBROUTINE AINPUT(AF,BF,AQ,BQ,AM,BM,AC,BC,P1,COVZ,AA,BA,N,NB,M,
     1NNB,KAI)
      DIMENSION AF(N,N),BF(NB,NB),AQ(N,N),BQ(NB,NB),AM(M,N),BM(M,NB),
     1AC(M,M),P1(N,N),COVZ(NNB,NNB),BC(M,M),AA(N,N),BA(NB,N)
    1 FORMAT(8E10.5)
    2 FORMAT(1X,12E11.4)
    3 FORMAT(/*  AF*)
    4 FORMAT(/*  BF*)
    5 FORMAT(/*  AQ*)
    6 FORMAT(/*  BQ*)
    7 FORMAT(/*  AM*)
    8 FORMAT(/*  BM*)
    9 FORMAT(/*  AC*)
   12 FORMAT(/*  BC*)
   10 FORMAT(/*  P1*)
   11 FORMAT(/*  COVZ*)
   13 FORMAT(/*  AA*)
   14 FORMAT(/*  BA*)
      DO 20 I=1,N
   20 READ 1, (AF(I,J),J=1,N)
      PRINT 3
      CALL DRUK(AF,N,N)
      DO 21 I=1,NB
   21 READ 1, (BF(I,J),J=1,NB)
      PRINT 4
      CALL DRUK(BF,NB,NB)
      PRINT 5
      GO TO(22,23)KAI
   22 CALL AOI(AQ,N,N, 1)
      READ 1,(AQ(I,I),I=1,N)
      PRINT2,(AQ(I,I),I=1,N)
      PRINT 6
      CALL AOI(BQ,NB,NB, 1)
      READ 1,(BQ(I,I),I=1,NB)
      PRINT2,(BQ(I,I),I=1,NB)
      GO TO 29
   23 DO 24 I=1,N
   24 READ 1,(AQ(I,J),J=1,I)
      DO 25 I=2,N
      II=I-1
      DO 25 J=1,II
   25 AQ(J,I)=AQ(I,J)
      CALL DRUK(AQ,N,N)
      PRINT 6
      DO 26 I=1,NB
   26 READ 1,(BQ(I,J),J=1,I)
      DO 27 I=2,NB
      II=I-1
      DO 27 J=1,II
   27 BQ(J,I)=BQ(I,J)
      CALL DRUK(BQ,NB,NB)
```

```
 29   DO 30 I=1,M
 30   READ 1, (AM(I,J),J=1,N)
      PRINT 7
      CALL DRUK(AM,M,N)
      DO 31 I=1,M
 31   READ 1, (BM(I,J),J=1,NB)
      PRINT 8
      CALL DRUK(BM,M,NB)
      DO 32 I=1,M
 32   READ 1, (AC(I,J),J=1,M)
      PRINT 9
      CALL DRUK(AC,M,M)
      DO 33 I=1,M
 33   READ 1, (BC(I,J),J=1,M)
      PRINT 12
      CALL DRUK(BC,M,M)
      PRINT 10
      GO TO(35,40)KAI
 35   CALL AOI(P1,N,N, 1)
      READ 1,(P1(I,I),I=1,N)
      PRINT2,(P1(I,I),I=1,N)
      CALL AOI(COVZ,NNB,NNB, 1)
      READ 1,(COVZ(I,I),I=1,NNB)
      PRINT 11
      PRINT2,(COVZ(I,I),I=1,NNB)
      GO TO 49
 40   DO 41 I=1,N
 41   READ 1,(P1(I,J),J=1,I)
      DO 43 I=1,NNB
 43   READ 1,(COVZ(I,J),J=1,I)
      DO 45 I=2,N
      II=I-1
      DO 45 J=1,II
 45   P1(J,I)=P1(I,J)
      CALL DRUK(P1,N,N)
      DO 47 I=2,NNB
      II=I-1
      DO 47 J=1,II
 47   COVZ(J,I)=COVZ(I,J)
      PRINT 11
      CALL DRUK(COVZ,NNB,NNB)
 49   CALL AOI(AA,N,N, 1)
      READ  1,(AA(I,I),I=1,N)
      PRINT 13
      PRINT 2,(AA(I,I),I=1,N)
      DO 50 I=1,NB
 50   READ  1, (BA(I,J),J=1,N)
      PRINT 14
      CALL DRUK(BA,NB,N)
      RETURN  $  END
```

```
      SUBROUTINE MEXP(D,F,A,T1,T2,EXPTA,T3,N,KMAX,AE,RE,KLM)
      DIMENSION A(N,N),F(N,N),EXPTA(N,N),T1(N,N),T2(N,N),T3(N,N)
      DOUBLE PRECISION T1,T2,EXPTA,PK,T,TK
    2 FORMAT(*   NUMBER OF ITERATIONS  K=*I4)
    5 FORMAT(/*  MAX. NUMBER OF ITERATIONS EXCEEDED*)
      T=D
      PK=1.D
      DO 10 J=1,N
      DO 10 I=1,N
      T1(I,J)=T*A(I,J)
   10 EXPTA(I,J)=T1(I,J)
      GO TO (14,18)  KLM
   14 DO 15 I=1,N
   15 EXPTA(I,I)=EXPTA(I,I)+1.0D
   18 PK=PK+1.D
      TK=T/PK
      DO 30 I=1,N
      DO 30 J=1,N
      T2(I,J)=0.D
      DO 30 L=1,N
   30 T2(I,J)=T2(I,J)+F(I,L)*T1(L,J)
   36 DO 20 J=1,N
      DO 20 I=1,N
      GO TO(38,39)  KLM
   38 T1(I,J)=T2(I,J)*TK    $  GO TO 20
   39 T1(I,J)=(T2(I,J) + T2(J,I))*TK
   20 EXPTA(I,J)=EXPTA(I,J) + T1(I,J)
   68 DO 70 J=1,N
      DO 70 I=1,N
      ER=DABS(T1(I,J)/(AE+RE*DABS(EXPTA(I,J))))
      IF(ER.GT.1.0) GO TO 80
   70 CONTINUE
      GO TO  110
   80 IF(PK.LT.KMAX) GO TO 18
      PRINT 5    $  STOP
  110 K=PK
      PRINT 2,K
      DO 120 I=1,N
      DO 120 J=1,N
  120 T3(I,J)=EXPTA(I,J)
      CALL DRUK(T3,N,N)
      RETURN  $  END

      SUBROUTINE ERCOV(PHI,BPHI,RK,BRK,AM,BM,AK,AC,BC,P1,OH,COVU,COVZ,AA
     1,BA,TA,TC,N,NB,M,NNB,KT,KTF,IOUT,T)
      DIMENSION PHI(N,N),BPHI(NB,NB),TA(NB,NB),TC(N,NB)
     1,RK(N,N),BRK(NB,NB),AM(M,N),BM(M,NB),AK(N,M),AC(M,M),BC(M,M)
     2,AA(N,N),BA(NB,N),OH(NNB,NNB),COVU(NNB,NNB),COVZ(NNB,NNB),P1(N,N)
      COMMON T1(25,25),T2(25,25),T3(25,25),T4(25,25),T8(25,25),T5(50,50)
     1,T7(50,25),T9(50,25),T6(50,50)
```

```
      EQUIVALENCE (T1,T5),(T6,T7,T8)
    2 FORMAT(///*    MEASUREMENT NUMBER*I4)
    3 FORMAT(///*    TIME =*F11.4)
C     MAKE AA=AA-I
      DO 5 I=1,N
    5 AA(I,I)=AA(I,I)-1.
      DO 25 II=1,KTF
      TIME=T*FLOAT(II)*FLOAT(KT)
      CALL AOI(OH,NNB,NNB, 1)
      CALL AOI(COVU,NNB,NNB, 1)
      DO 10 I=1,NB
      DO 10 J=1,NB
      OH(I,J)=BPHI(I,J)
   10 COVU(I,J)=BRK(I,J)
      DO 15 I=1,N
      DO 15 J=1,N
   15 OH(I+NB,J+NB)=PHI(I,J)
      DO 18 K1=1,KT
C     UPDATE P1
      CALL ABTAC(PHI,P1,PHI,T1,T2,N,N,N,N, 2)
      CALL MATADD(T2,RK,P1,N,N,N)
C     UPDATE COVARIANCE(Z)
      CALL ABTAC(OH,COVZ,OH,T5,T6,NNB,NNB,NNB,NNB, 2)
      CALL MATADD(T6,COVU,COVZ,NNB,NNB,NNB)
      TIMEA=TIME-T*FLOAT(KT-K1)
      PRINT 3,TIMEA
   17 CALL OUT(P1,COVZ,T1,N,NB,NNB,IOUT)
   18 CONTINUE
      PRINT 2,II
      CALL FILTER(AM,AK,P1,AC,N,M)
      CALL MATMPY(AK,BM,T8,N,N,M,M,NB)
      CALL MATMPY(BA,T8,TA,NB,NB,N,N,NB)
      CALL MATMPY(AK,AM,TC,N,N,M,M,N)
      DO 65 I=1,NB
   65 TA(I,I)=TA(I,I)-1.
      DO 70 I=1,N
      TC(I,I)=TC(I,I)-1.
      DO 70 J=1,N
   70 TC(I,J)=-TC(I,J)
      CALL MATMPY(BA,TC,T2,NB,NB,N,N,N)
      CALL MATMPY(AA,TC,T4,N,N,N,N,N)
      CALL MATMPY(AA,T8,TC,N,N,N,N,NB)
      CALL SUM(TA,T2,TC,T4,OH,NB,N,NB,N,NNB,NNB)
      CALL ABTAC(AK,BC,AK,T2,T1,N,M,M,N, 2)
      CALL SUMB(BA,,AA,,T7,NB,N,N,,NNB,N)
      CALL ABTAC(T7,T1,T7,T9,COVU,NNB,N,N,NNB, 2)
      CALL ABTAC(OH,COVZ,OH,T5,T6,NNB,NNB,NNB,NNB, 2)
      CALL MATADD(T6,COVU,COVZ,NNB,NNB,NNB)
      CALL OUT(P1,COVZ,T1,N,NB,NNB,IOUT)
   25 CONTINUE
      RETURN $ END
```

```
      SUBROUTINE FILTER(AM,AK,P1,AC,N,M)
      DIMENSION AM(M,N),AK(N,M),P1(N,N),AC(M,M)
      COMMON T1(25,25),T2(25,25),T3(75,50)
    9 FORMAT(/*  AK*)
C     CALCULATE K
   20 CALL ABTAC(AM,P1,AM,T1,T2,M,N,N,M, 2)
      CALL MATADD(T2,AC,T1,M,M,M)
      NO=0
      CALL MATINV(T1,M,T1,NO,DET,M)
      CALL  ABT(P1,AM,T2,N,N,M)
      CALL  MATMPY(T2,T1,AK,N,N,M,M,M)
C     CALCULATE P1
      CALL ABTAC(AK,AM,P1,T1,T2,N,M,N,N, 3)
      CALL MATSUB(P1,T2,P1,N,N,N)
   40 PRINT 9
      CALL DRUK(AK,N,M)
      RETURN  $  END

      SUBROUTINE OUT(P1,COVZ,T,N,NB,NNB,IOUT)
      DIMENSION P1(N,N),COVZ(NNB,NNB),T(NB,NB)
    2 FORMAT(/*  COVZ*)
    3 FORMAT(/*  SQUARE ROOT OF UPPER DIAGONAL TERMS OF COVZ.*)
    5 FORMAT(1X,12E11.4)
    8 FORMAT(/*  P1*)
      IF(IOUT.LE.1) GO TO 19
      PRINT 8
      CALL DRUK(P1,N,N)
      PRINT 2
      CALL DRUK(COVZ,NNB,NNB)
   19 PRINT 3
      DO 20 I=1,NB
   20 T(I,I)=SQRTF(COVZ(I,I))
      PRINT 5,(T(I,I),I=1,NB)
      RETURN  $  END

      SUBROUTINE DRUK(A,N,M)
      DIMENSION A(N,M)
  101 FORMAT(*    TRANSPOSE*)
  102 FORMAT(1X,12E11.4)
      IF(N.GT.M) GO TO 2
      DO 1 I=1,N
    1 PRINT102,(A(I,J),J=1,M)
      RETURN
    2 PRINT 101
      DO 3 I=1,M
    3 PRINT 102,(A(J,I),J=1,N)
      RETURN  $  END
```

```
      SUBROUTINE SUM(A,B,C,D,T,I1,I2,J1,J2,I12,J12)
      DIMENSION A(I1,J1),B(I1,J2),C(I2,J1),D(I2,J2),T(I12,J12)
      DO 20 J=1,J2
      DO 15 I=1,I1
   15 T(I,J+J1)=B(I,J)
      DO 20 I=1,I2
   20 T(I+I1,J+J1)=D(I,J)
      ENTRY SUMB
      DO 10 J=1,J1
      DO  5 I=1,I1
    5 T(I,J)=A(I,J)
      DO 10 I=1,I2
   10 T(I+I1,J)=C(I,J)
      RETURN  $  END

      SUBROUTINE AOI(A,N,M, KK)
      DIMENSION  A(N,M)
      DO 10 I=1,N
      DO 10 J=1,M
   10 A(I,J)=0.0
      IF(KK.EQ.1) RETURN
      DO 20 I=1,N
   20 A(I,I)=1.0
      RETURN  $  END

      SUBROUTINE ABT(A,B,T,K,L,M)
      DIMENSION A(K,L),B(M,L),T(K,M)
      DO 1 J=1,M
      DO 1 I=1,K
      T(I,J)=0.0
      DO 1 K1=1,L
    1 T(I,J)=T(I,J)+A(I,K1)*B(J,K1)
      RETURN  $  END

      SUBROUTINE ABTAC(A,B,C,T,Y,K,L,M,N,KK)
      DIMENSION A(K,L),B(L,M),C(N,M),T(K,M),Y(K,N)
C     ABAT   KK=2  Y=A*B*AT
C     ABCT   KK=3  Y=A*B*CT
      CALL MATMPY(A,B,T,K,K,L,L,M)
      GO TO(9,2,5) KK
    2 DO 4 I=1,K
      DO 4 J=1,N
      Y(I,J)=0.0
      DO 3 K1=1,M
    3 Y(I,J)=Y(I,J)+T(I,K1)*C(J,K1)
    4 Y(J,I)=Y(I,J)
      GO TO 9
    5 CALL ABT(T,C,Y,K,M,N)
    9 RETURN  $  END
```

BIBLIOGRAPHY

[1] Bongiovanni, P. L., Comparative Analysis of Three Estimation Techniques, M.S.E.E. Thesis, University of Massachusetts, June 1967.

[2] Kalman, R. E., A New Approach to Linear Filtering and Prediction Problems, ASME Trans. (Journal of Basic Engineering), Vol. 82D, March 1960, pp. 35-45.

[3] Kalman, R. E. and Bucy, R. S., New Results in Linear Filtering and Prediction Theory, ASME Trans. (Journal of Basic Engineering), Vol. 83D, March 1961, pp. 95-108.

[4] Tou, J. T., Modern Control Theory, McGraw-Hill Book Company, New York, 1964.

[5] Bryson, A. E., Jr. and Ho, Y. C., Optimum Programming, Estimation and Control (to be published).

[6] Cunningham, S. L., Fairbanks, R. L. and Allione, M. S., Linear Systems Analysis Program, "LISAP," Autonetics Calif. Report No. T6-220/3111, March 1966.

[7] Pitman, G. R., Jr., Inertial Guidance, Wiley & Sons, New York, 1962.

[8] Hutchinson, C. E. and Monopoli, R. V., Optimum Use of Reference Information in Inertial Navigation, Preprints of Technical Papers, 18th Annual National Aerospace Electronics Conference, Dayton, Ohio, 1966, pp. 195-197.

[9] Frazer, R. A., Duncan, W. J. and Collar, A. R., Elementary Matrices, Cambridge, University Press, 1938, p. 42.

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

School of Engineering
University of Massachusetts
Amherst, Massachusetts 01002

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

An Error Analysis Technique for Inertial Navigation Systems and Kalman Filters

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

C. E. Hutchinson and H. M. Wondergem

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September, 1968	79	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
ONR-N00014-68-A-0146	THEMIS-UM-68-2
b. PROJECT NO ONR-N00014-68-A-0146-6	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Department of Defense (Project THEMIS)

13. ABSTRACT

This report develops a method for analyzing the errors incurred in optimum estimation schemes. The optimum control law for modern control systems is usually a function of all the states of the system. In many practical cases it is necessary to extract information concerning the states from the measurement of the output. This extraction process is called estimation. Due to random measurement noise and the random noise components of the states of the system, the estimate of the states is not perfect.

A set of computer programs has been developed that provides a way of evaluating optimum and sub-optimum Kalman estimation techniques. The random noise processes have been described in a statistical manner, in order to specify the random noise quantitatively as a function of time.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

UNCLASSIFIED

Security Classification

A-31409

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Error Analysis						
Navigation						
Inertial Navigation						
Kalman Filter						
Optimum						